

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Makoto NAKANISHI

Application No.: NEW

Group Art Unit: Not Yet Assigned

Filed: March 11, 2004

Examiner: Not Yet Assigned

For: MATRIX PROCESSING DEVICE IN SMP NODE DISTRIBUTED MEMORY THPE
PARALLEL COMPUTER

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN
APPLICATION IN ACCORDANCE
WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55**

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith
a certified copy of the following foreign application:

Japanese Patent Application No(s). 2003-95720

Filed: March 31, 2003

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing
date(s) as evidenced by the certified papers attached hereto, in accordance with the
requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: March 11, 2004

By: 

J. Randall Beckers
Registration No. 30,358

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

JAPAN PATENT OFFICE

This is to certify that the annexed is a true copy of the following application as filed with this Office.

Date of Application: March 31, 2003

Application Number: Patent Application
No. 2003-095720

[ST.10/C]: [JP2003-095720]

Applicant(s) : FUJITSU LIMITED

December 9, 2003

Commissioner,
JAPAN Patent Office Yasuo IMAI

Certificate No. P2003-3101648

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 2003年 3月31日
Date of Application:

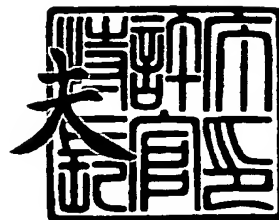
出願番号 特願2003-095720
Application Number:
[ST. 10/C]: [JP 2003-095720]

出願人 富士通株式会社
Applicant(s):

2003年12月 9日

特許庁長官
Commissioner,
Japan Patent Office

今井 康



出証番号 出証特2003-3101648

【書類名】 特許願

【整理番号】 0350107

【提出日】 平成15年 3月31日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 17/12

【発明の名称】 S M P ノード分散メモリ型並列計算機における行列処理装置

【請求項の数】 5

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 中西 誠

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100074099

【住所又は居所】 東京都千代田区二番町8番地20 二番町ビル3F

【弁理士】

【氏名又は名称】 大菅 義之

【電話番号】 03-3238-0031

【選任した代理人】

【識別番号】 100067987

【住所又は居所】 神奈川県横浜市鶴見区北寺尾7-25-28-503

【弁理士】

【氏名又は名称】 久木元 彰

【電話番号】 045-573-3683

【手数料の表示】

【予納台帳番号】 012542

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9705047

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 SMP ノード分散メモリ型並列計算機における行列処理装置

【特許請求の範囲】

【請求項 1】 複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの 1 巻き分を、該 1 巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第 1 の配置ステップと、

該 1 巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを 1 次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第 2 の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列に LU 分解する LU 分解ステップと、

LU 分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法を情報装置に実現させるプログラム。

【請求項 2】 前記 LU 分解は、再帰的手続きにより、各ノードの各プロセッサで並列的に行われることを特徴とする請求項 1 に記載のプログラム。

【請求項 3】 前記更新ステップにおいては、各ノードが、列ブロックを計算している間に、計算し終わった部分のデータであって、他のブロックの更新に必要なデータを該計算と平行して他のノードに転送することを特徴とする請求項 1 に記載のプログラム。

【請求項 4】 前記並列計算機は、SMP (Symmetric Multi Processor) を各ノードとする SMP ノード分散メモリ型並列計算機であることを特徴とする請求項 1 に記載のプログラム。

【請求項 5】 複数のプロセッサとメモリを含む複数のノードをネットワーク

で接続した並列計算機における行列処理装置であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの1巻き分を、該1巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第1の配置手段と、

該1巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離手段と、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを1次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第2の配置手段と、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にLU分解するLU分解手段と、

LU分解されたブロックを用いて、行列の他のブロックを更新する更新手段と、
を備えることを特徴とする行列処理装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、SMP (Symmetric MultiProcessor) ノード分散メモリ型並列計算機における行列処理装置あるいは処理方法に関する。

【0002】

【従来の技術】

ベクトルプロセッサをクロスバーで結合した並列計算機向けに開発した連立一次方程式の解法では、ブロックLU分解の各ブロックを各PEにサイクリックに配置してLU分解を行っていた。ベクトルプロセッサではブロック幅を小さくしてもコストの高い行列積による更新部分の計算効率は非常に高かった。このためブロック幅12程度でサイクリックな配置と見なして、まず、このブロックをLU分解及び1つのCPUで逐次的に計算してから、結果を部分的に分割して各プロセッサに転送して、行列積での更新を行っていた。

【0003】

図 26 は、スーパースカラ並列計算機用 LU 分解法のアルゴリズムを概略説明する図である。

配列 A を外積形式のガウスの消去法をブロック化した方法で LU 分解する。ブロック幅 d で分解する。

k 番目の処理で、更新部分 $A^{(k)}$ を次の計算で更新する。

$$A^{(k)} = A^{(k)} - L_2^{(k)} \times U_2^{(k)} \dots \dots (1)$$

k+1 番目の処理では、 $A^{(k)}$ を幅 d で分解して d だけ小さいマトリックスを同じ式で更新する。

$L_2^{(k)}$ 、 $U_2^{(k)}$ は以下の式で求める必要がある。

式 (1) で更新を行う場合、

【0004】

【数 1】

$$\bar{A}^{(k)} = (L_1^{(k)T}, L_2^{(k)T})^T U_1^{(k)}$$

【0005】

と分解し、 $U_2^{(k)} = L_1^{(k)-1} U_2^{(k)}$ と更新する。

上記のブロック化された LU 分解の方法は、特許文献 1 に記載されている。

そのほか、並列計算機で行列を計算する技術として特許文献 2 には、連立 1 次方程式の係数行列を外部記憶装置に格納する方式が、特許文献 3 には、ベクトル計算機における方式が、特許文献 4 には、多枢軸同時消去を行う方式が、特許文献 5 には、スパース行列の各要素の構成を並び替えて、縁付きブロック対角行列にしてから LU 分解を行う方法が記載されている。

【0006】

【特許文献 1】

特開 2002-163246 号公報

【特許文献 2】

特開平 9-179851 号公報

【特許文献 3】

特開平 11-66041 号公報

【特許文献 4】

特開平 5 - 2 0 3 4 9 号公報

【特許文献 5】

特開平 3 - 2 2 9 3 6 3 号公報

【0 0 0 7】

【発明が解決しようとする課題】

上記スーパスカラ並列計算機用 L U 分解の方法を単純に一つのノードを S M P とする並列計算機システムで行うと以下の問題が発生する。

【0 0 0 8】

S M P ノードでの行列積を効率的に行うためにはベクトル計算機で 1 2 と設定していたブロック幅を 1 0 0 0 程度に増やす必要がある。

(1) この結果、ブロック毎にそれが各プロセッサにサイクリックに配置されていると見なして処理を行うと、行列積での更新の計算量がプロセッサ間で不均一である割合が大きくなり並列効率が著しく低下する。

(2) また、1 ノードで計算する幅 1 0 0 0 程度のブロックの L U 分解は、ノード内でのみ計算すると、他のノードはアイドル状態となる。幅の大きさに比例して、このアイドル時間が増えるため、並列化効率が著しく低下する。

(3) S M P ノードを構成する C P U 数を増やすと計算能力の増加に対して、転送スピードが相対的に劣化しているため、従来の方法は転送量が約 $0.5n^2 \times 1.5$ 要素（ここでの要素は、行列の要素である）であったが、相対的に増えて見える。このため効率がかなり落ちる。

(1) ~ (3) までの劣化は全体で焼く 2 0 ~ 2 5 % の性能ダウンを引き起こす。

【0 0 0 9】

本発明の課題は、S M P ノード分散メモリ型並列計算機で高速に行列を処理することの出来る装置あるいは方法を提供することである。

【0 0 1 0】

【課題を解決するための手段】

本発明の行列処理方法は、複数のプロセッサとメモリを含む複数のノードをネ

ットワークで接続した並列計算機における行列処理方法であって、ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの1巻き分を、該1巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第1の配置ステップと、該1巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを1次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第2の配置ステップと、該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にLU分解するLU分解ステップと、LU分解されたブロックを用いて、行列の他のブロックを更新する更新ステップとを備えることを特徴とする。

【0011】

本発明によれば、各ノード間の計算負荷を分散し、並列化度を上げることが出来るので、より高速な行列処理が行える。また、演算と、データ転送を並列して行うことから、計算機の処理能力をデータ転送のスピードに制限されずに、向上することが出来る。

【0012】

【発明の実施の形態】

本発明の実施形態においては、ブロック幅を大きくしても負荷バランスが完全に均一であり、1CPUで逐次計算していた部分をノード間で並列に処理する方式を提案する。

【0013】

図1は、本発明の実施形態が適用されるSMPノード分散メモリ型並列計算機の概略全体構成を示す図である。

図1(a)に示されるように、クロスバーネットワークにノード1～ノードNが接続され、相互に通信できるようになっている。各ノードは、図1(b)に示されるように相互結合網10によって、メモリモジュール11-1～11-n、及びプロセッサ13-1～13-mとキャッシュ12-1～12-mの組とが相互に結合され、通信可能となっている。データ通信用ハード(DTU)14は、

図1 (a) のクロスバーネットワークに接続され、他のノードと通信可能となっている。

【0014】

まず、比較的ブロック幅の小さなコラムブロックをノードにサイクリックに配置する。各ノードに一巻き分（サイクリックにコラムブロックを配置した場合の1回でサイクリックに配置される分）あるブロックを一つに束ねたものを一つに行列と見なす。これは行列を2次元目を均等に分割し、各ノードに分散配置した状態と見なすことが出来る。これを1次元目を均等に分割した配置に並列転送を利用して動的に変更する。ここで、1次元目を分割、2次元目を分割とは、行列を長方形あるいは正方形とした場合、横方向を縦の線で分割することを1次元目を分割すると言い、縦方向を横の線で分割することを2次元目を分割するという。このとき一番上の正方形部分は各ノードが重複して持つようにする。

【0015】

この分散配置の変更でクロスバーネットワークを利用した並列転送が使える、転送量はノード数分の1となる。1次元目を均等に分割した配置に変更した配置で、ノード間通信を使って、このブロックのLU分解を並列に行う。このとき並列化効率があがり、かつSMPの性能を引き出せるようにするために、更にブロックに分解して再帰的なLU分解を行う。

【0016】

このブロックLU分解が終了した時点で各ノードには対角ブロック部分の情報と1次元目を均等に分割した部分の情報があるため、これを利用して行ブロック部分を更新して、保持している列ブロック部分とで更新できる部分を更新する。更新時に隣のノードにこの情報を転送して、次の更新の準備を行う。この転送は計算と同時に出来る。これらの操作を繰り返して全ての更新部分の更新を行う。

【0017】

図2は、本発明の実施形態に従った全体の処理フローチャートである。

まず、ステップS10において、最後の一巻きか否かを判断する。ステップS10の判断がYESの場合には、ステップS15に進む。ステップS10の判断がNOの場合には、ステップS11において、対象となる一巻き分のブロックを

結合したブロックを1次元目で分割した配置に並列転送を利用して変換する。このとき対角ブロックは全てのノードで共通に持つようにする。ステップS12においては、1次元目を分割配置したブロックに関してLU分解を行う。このときキャッシュの大きさを考慮したブロック幅までと、そのブロック幅より小さい部分の処理を再帰的な手続きで行う。ステップS13では、LU分解した1次元目で分割配置されたブロックを並列転送をつかって元の2次元目を分割した配置に戻す。ステップS14においては、この時点で各ノードには対角ブロックと残りをノード数に1次元目で分割した小ブロックが各ノードに割り付けられている。各ノードで共通に持っていた更新済みの対角ブロックを使ってブロック行を各ノードで更新する。このとき次の更新で必要となる列ブロックを隣のノードに計算と同時に転送する。ステップS15では、最後の一巻きは各ノードに分割せずに冗長に配置して、同じ計算を行ってLU分解を行う。各ノード部分に対応する部分をコピーバックする。そして、処理を終了する。

【0018】

図3は、本発明の実施形態の一般概念図である。

図3に示されるように、行列を例えば、4等分して各ノードに分散配置する。各ノードは、列ブロックが割り当てられており、サイクリックな順序で処理する。このとき一巻き分を束ねて1つのブロックと見なす。これを対角ブロック部分を除き1次元目で分割し、通信を使って各ノードに再配置する。

【0019】

図4及び図5は、比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図である。

図4及び図5に示すように、行列の一部の列ブロックを、更に小さい列ブロックに細分化し、各ノード（今の場合4つとしている）にサイクリックに割り当てる。このような配置の変更は、2次元目を分割されたブロックを1次元目を分割（対角ブロックは共通に保持）変更することになる。これはクロスバーネットワークの並列転送を利用して変更することが出来る。

【0020】

これは、1巻きが結合されたブロックをメッシュに仮想的に分割したとき、対

角線方向のブロックの並び (1 1、2 2、3 3、4 4)、(1 2、2 3、3 4、4 1)、(1 3、2 4、3 1、4 2)、(1 4、2 1、3 2、4 3) の各組のブロックを各ノードに (二次元目の示すプロセッサから 1 次元目の示すプロセッサに転送する) 並列転送することで実現できる。このとき、対角ブロック部分も一緒に送ることで対角ブロック部分は各ノードが共通に持つことができる十分な大きさで、転送はプロセッサ数分の 1 になる。

【0021】

このように分散配置を変更した列ブロックに対する LU 分解を、各ノードに対角ブロックと残りの部分を均等に分割したものを配置して、ノード間通信及びノード間で同期を取りながら処理を行う。また、ノード内での LU 分解の処理はスレッド並列化を行う。

【0022】

スレッド並列化での LU 分解がキャッシュ上で効率的に行えるように、2 重構造の再帰的手続で行う。つまり、あるブロック幅までの大きさで一次の再帰手続で行い、それより小さい部分に関しては、スレッド並列化のために、各スレッドで、そのブロックを対角部分と残りの部分を並列処理するスレッド数で均等に分割した部分を合わせて連続な作業域にコピーして処理を行う。このことでキャッシュ上のデータを有効に利用する。

【0023】

また、ノード間で共有している対角ブロック部分の計算はノード間で冗長に計算されてノード間の LU 分解の並列化効率が劣化する。LU 分解を 2 重の再帰的手続きで行うことで、各ノード内でスレッドで並列計算するときのオーバーヘッドを減らすことが出来る。

【0024】

図 6 は、図 4 及び図 5 で配置されたブロックの更新処理を説明する図である。

図 6 の最も左のブロックは各ノードに対角ブロックを冗長に、かつ、残りのブロックを一次元目で均等に分割したものを作業域に配置したものである。あるノードでの状態と考える。最小ブロック幅まで 1 次の再帰手続きを行う。

【0025】

最小ブロックのLU分解が終わったら、この情報を使って、行ブロック及び更新部分の更新を更新する領域を均等に分割して、並列に更新する。

最小ブロック部分のLU分解は、更に以下のように最小幅のブロックの対角部分を共通に、かつ、残り部分を均等に分割して、各スレッドの局所領域（キャッシュの大きさ程度）にコピーする。

【0026】

この領域を使って、更に再帰的手続きでLU分解を行う。ピボットを決めて、行の入れ替えを行うために各スレッドに、ピボットの相対的位置から、ノードでの相対位置、全体での位置に換算するための情報を保持しておく。

【0027】

ピボットがスレッドの局所領域の対角部分内にあるときは、各スレッドで、独立に入れ替えを行える。

スレッドの対角ブロックを超えたときは、その位置が、以下の条件のときによって処理が異なる。

a) ピボットがノード間に分割配置したとき冗長に配置した対角ブロック内にあるとき。

【0028】

このときは、ノード間で通信する必要はなく、各ノードで独立に処理できる。

b) ピボットがノード間に分割配置した時冗長に配置した対角ブロックを超えたとき。

【0029】

このときはスレッド間での最大値、つまりノードでの最大値を全ノードに通信して最大ピボットがどのノードに有るかを決定する。これが決まった後、最大ピボットを持つノードで行の入れ替えを行う。そのあと、入れ替えられた行（ピボット行）を他のノードに通信する。

【0030】

このようなピボットの処理を行う。

2重構造を持つ再帰手続きでのLU分解の二次のスレッド並列で行うLU分解は、上記のピボット処理を行いながら、各スレッドの局所領域でLU分解を並列

に行うことができる。

【0031】

ピボットの入れ替えの履歴は共用メモリに各ノードに冗長に保持する。

図7は、再帰的なLU分解の手順を説明する図である。

再帰的なLU分解の手順は以下のようになる。

【0032】

図7(b)のレイアウトを考える。図7(b)の対角ブロック部分がLU分解できると、UはL1を使って、 $U \leftarrow L^{-1}U$ 、 $C \leftarrow L \times U$ と更新する。

再帰の手順は、LU分解する領域を前半と後半に分割し、分割した領域をLU分解の対象と見なして、再帰的に行う方法である。ブロックの幅が、ある最小の幅より小さくなったとき、その幅に関しては従来通りのLU分解を行う。

【0033】

図6(a)は、領域を真ん中の太線で2分割し、その左側をLU分解する過程で更に2分割したところである。太線で分割した左側は図6(b)のレイアウトを当てはめられる。このレイアウトのCの部分もLU分解できたとき、太線から左側のLU分解が終わる。

【0034】

この左側の情報から、図6(b)のレイアウトを全体にあてはめて、Cとなる右側の更新を行う。更新が終わったら、右側に図6(b)のレイアウトを当てはめて同じようにLU分解を行う。

・ブロックのLU分解処理の後の行の入れ替えと行ブロックの更新及びrank p updateでの更新

ノード間にブロックを再配置した状態でノード間通信及びスレッド並列を使ってLU分解を並列に実行した後、各ノードには各ノードに共通に置かれた対角ブロックと残りの部分を均等に分割した部分のひとかけらがLU分解された値を保持して残る。

【0035】

各ノードでピボットの入れ替えの履歴の情報と対角ブロックの情報を使って、まず行の入れ替えを行う。その後、行ブロック部分の更新を行う。この後、対角

ブロックの残り部分を分割した列ブロック部分と更新された行ブロック部分を利用して更新部分を更新する。この計算と同時に更新に使う分割された列ブロック部分を全ノードで隣のノードに転送する。

【0036】

この転送は、次の更新で必要な情報を計算と同時に送り、次の計算の前までに準備を行うためであり、転送を計算と同時に行うことで計算を効率よく続けることができる。

【0037】

また、部分的な行列積の更新をスレッド数が多くても効率的に行えるように各スレッドで計算する行列積の更新領域が正方形に近くなる用に分割する。各ノードで更新を受け持つ更新領域は、正方形である。この領域の更新を各スレッドに分担して、かつ、性能劣化を引き起こさないようにすることを考える。

【0038】

このため、更新領域をできるだけ正方形に近い形に分割する。このことで更新部分の2次元目の大きさがかなり大きく取れ、行列積の計算で繰り返し参照される部分の参照をキャッシュ上に保持して有効利用することが比較的できるようになる。

【0039】

このために、以下の手順で行列積の更新の各スレッドでの分担を決めて並列計算する。

- 1) スレッドの総数 $\# \text{THRD}$ の平方根を求める。
- 2) この値が整数でないとき、これを切り上げて $nrow$ とする。
- 3) 2次元目の分割数を $nrow$ とする。
- 4) 1次元目の分割数を $ncol$ を以下の条件を満たす最小の整数を見つける。
 $ncol \times nrow \geq \# \text{THRD}$
- 5) if($ncol * nrow == \# \text{thrd}$) then

1次元目を $ncol$ 等分、2次元目を $nrow$ 等分 $ncol * nrow$ に分割して各スレッドに更新を並列実行させる。

else

1次元目を $ncol$ 等分、2次元目を $nrow$ 等分して $ncol * nrow$ に分割して (1, 1)、(1, 2)、(1, 3)、 \dots (2, 1)、(2, 2)、(2, 3) \dots と #THRD 個の部分を並列更新する。残りの領域は一般的に横に長い長方形となる。これを2次元目を均等に分割して全スレッドで負荷が均等になるように更新部分を分割して再度並列処理する。

endif

・ソルバー部分

図8は、対角部分以外の部分ブロックの更新について説明する図である。

【0040】

LU分解された結果は、各ノードに分散配置された形で保存されている。各ノードには比較的ブロック幅の小さなブロックがLU分解された状況で格納されている。

【0041】

この幅の小さなブロックに関して前進代入、後退代入を行って次のブロックのある隣のノードに処理を渡す。このとき解を更新した部分を隣のノードに転送する。

【0042】

実際の前進代入及び後退代入では細長いブロックで対角ブロック部分を除いた長方形部分を1次元目で均等にスレッド数で分割して並列更新を行う。

まず、一つスレッドで $LD \times BD = BD$ を解く。

【0043】

この情報を使って全スレッドで以下のように B を並列に更新する。

$$B_i = B_i - L_i \times BD$$

この1サイクルの更新で変更された部分を隣のノードに転送する。

【0044】

前進代入が終わったら、今までの処理でノードに処理を渡してきたのとちょうど逆を辿るようにして後退代入を行う。

実際には、元の行列の各ノードに配置された部分をサイクリックに処理している。これは列ブロックを入れ替えて別の行列に変換していることに相当する。L

U分解の過程でピボットをとる列は未分解部分のどの列を対象にしてもよいことに由来する。

$AP P^{-1}x = b \rightarrow y = P^{-1}x$ と置いて y について解くことに相当する。解いた y を並び変えることで x を求めることが出来る。

【0045】

図9～図11は、行ブロックの更新処理を説明する図である。

列ブロックの計算が終わったら、今度計算された部分をもとの2次元目を分割した配置に戻す。ここで、2次元目を分割した形でのデータは各ノードに保持しておく。次に、行の入れ替え情報を元に、行の入れ替えを行ったあと、行ブロックを更新する。

【0046】

各ノードに存在する列ブロックの部分を計算と同時に隣のノードにリング状に送ることで順次更新を進めていく。バッファをもう一つ持つことで可能となる。この領域には各ノードに対角ブロックを冗長に保持しているが、これも一緒に転送する。対角ブロック以外の部分のデータの量が多く、また、計算と同時に転送を行うので、転送時間は見えない。

【0047】

図10によれば、バッファAからBへのデータ転送を行う。次のタイミングではバッファBからAへのノードのリングに沿ってデータを送る。このようにしてスイッチしてデータ送る。更に、図11において、更新が終わったら、列ブロックと行ブロックを除いた正方行列に対して大きさが縮小したものに対して同じ処理を繰り返す。

【0048】

図12～図25は、本発明の実施形態のフローチャートである。

図12及び図13はサブルーチンpLUのフローである。このサブルーチンは、呼び出しプログラムであり、各ノードで1つのプロセスを生成してから呼び出すことで並列に処理を行う。

【0049】

まず、解くべき問題の大きさを、単位ブロック数を $iblkunit$ 、ノード数を num

nordとして、 $n = \text{iblkunit} \times \text{numnord} \times m$ (m は各ノードでの単位ブロック数) としたLU分解を行う。各ノードに係数行列Aの2次元目を均等に分割した共用メモリA ($k, n / \text{numnord}$) ($k \geq n$) 及び行の入れ替えの履歴を格納するip(n)を引数として受け取る。ステップS 2 0において、nonordにプロセス番号 (1 ~ ノード数) を設定し、numnordにノード数 (全プロセス数) を設定する。ステップS 2 1において、各ノードでスレッドを生成し、nothrdにスレッド番号 (1 ~ スレッド数) 及びnumthrdにスレッドの総数を設定する。ステップS 2 2において、ブロック幅の設定である $\text{iblkmacro} = \text{iblkunit} \times \text{numnord}$ 、繰り返し回数である $\text{loop} = n / (\text{iblkunit} \times \text{numthrd}) - 1$ を計算し、更に、 $i = 1$ 、 $\text{lenbufmax} = (n - \text{iblkmacro}) / \text{numnord} + \text{iblkmacro}$ を設定する。

【 0 0 5 0 】

ステップS 2 3において、wlul ($\text{lenbufmax}, \text{iblkmacro}$)、wlu2($\text{lenbufmax}, \text{iblkmacro}$)、bufs($\text{lenbufmax}, \text{iblkunit}$)、bufd($\text{lenbufmax}, \text{iblkunit}$)の作業域を確保する。この領域をサブルーチンが実行の都度、実際の長さlenbufを計算して、必要な大きさだけ使う。

【 0 0 5 1 】

ステップS 2 4においては、 $i \geq \text{loop}$ であるか否かを判断する。ステップS 2 4の判断がYESの場合には、ステップS 3 7に進む。ステップS 2 4の判断がNOの場合には、ステップS 2 5において、ノード間でバリア同期を取る。そして、ステップS 2 6において、 $\text{lenblks} = (n - i \times \text{iblkmacro}) / \text{numnord} + \text{iblkmacro}$ を計算する。ステップS 2 7において、サブルーチンctobを呼び出し、各ノードにある幅iblkunitの i 番目を対角ブロックと1次元目を均等分割した幅iblkmacroの部録を対角ブロックに結合し、ノードに持つ配置を変える。ステップS 2 8では、ノード間でバリア同期を取る。ステップS 2 9では、サブルーチンinterluを呼び出して、配列wlulに格納され、分散再配置された、ブロックをLU分解する。行の入れ替えの情報は、 $\text{is} = (i - 1) \times \text{iblkmacro} + 1$ 、 $\text{ie} = i \times \text{iblkmacro}$ としてip(is:ie)に格納されている。

【 0 0 5 2 】

ステップS 3 0において、ノード間でバリア同期を取り、ステップS 3 1にお

いて、サブルーチンbtocを呼び出して、再配置されたブロックでLU分解されたブロックを各ノードのもとも格納されていた場所に戻す。ステップS32においてノード間でバリア同期を取り、ステップS33において、サブルーチンexrwを呼び出して、行の入れ替え及び行ブロックの更新を行う。ステップS34においては、ノード間でバリア同期を取り、ステップS35において、サブルーチンmmcbtを呼び出して、各ノードにある列ブロックの部分（wlulに格納されている）と行ブロックの部分との行列積で更新する。計算と同時に列ブロック部分をプロセッサ間をリングに沿って転送し、次の更新の準備を行いながら更新する。ステップS36においては、 $i=i+1$ として、ステップS24に戻る。

【0053】

ステップS37では、ノード間でバリア同期を取り、ステップS38において、生成したスレッドを消滅させる。ステップS39において、サブルーチンfbluを呼んで、最後のブロックのLU分解を行いながら更新する。ステップS40において、ノード間でバリア同期を取り、処理を終了する。

【0054】

図14及び図15は、サブルーチンctobのフローである。

ステップS45において、 $A(k, n/\text{numnord})$ 、 $\text{wlul}(\text{lenblks}, \text{iblksmacro})$ 、 $\text{bufs}(\text{lenblks}, \text{iblkunit})$ 、 $\text{bufd}(\text{lenblks}, \text{iblkunit})$ を引数で受けて、各ノードの*i*番目の幅iblkunitのブロックをnumnord個束ねたものの対角ブロック行列部分より下の部分をnumnord個に分割したものと対角ブロックを加えたものを各ノードに分散配置したものに転送を利用して配置換えする。

【0055】

ステップS46においては、 $\text{nbase}=(i-1)*\text{iblksmacro}$ （*i*は呼び出し元のメインループの繰り返し回数）、 $\text{ibs}=\text{nbase}+1$ 、 $\text{ibe}=\text{nbase}+\text{iblksmacro}$ 、 $\text{len}=(\text{n}-\text{ibe})/\text{numnord}$ 、 $\text{nbase2d}=(i-1)*\text{iblkunit}$ 、 $\text{ibs2d}=\text{nbase2d}+1$ 、 $\text{ibe2d}=\text{ibs2d}+\text{iblkunit}$ を計算する。ここで、送信データ数は $\text{lensend}=(\text{len}+\text{iblksmacro})*\text{iblkunit}$ である。ステップS47においては、 $\text{iy}=1$ と設定し、ステップS48において、 $\text{iy} > \text{numnord}$ か否かを判断する。ステップS48の判断がYESの場合には、サブルーチンを抜ける。ステップS48の判断がNOの場合には、ステップS49にお

いて、送信する部分、受信する部分を決める。すなわち、 $idst = \text{mod}(\text{nonord} - 1 + iy - 1, \text{numnord}) + 1$ (送信先ノード番号)、 $isrs = \text{mod}(\text{nonord} - 1 + \text{numnord} - iy + 1, \text{numnord}) + 1$ (送信元ノード番号) を計算する。ステップ S 50 においては、各ノードで自分に割り付いている幅 $iblk\text{sunit}$ の対角ブロック部分と、その下部分のブロックの 1 次元目を numnord で分割した部分で、再配置した時保持する部分 (転送先のノード数番目のもの) をバッファの下部分に格納する。すなわち、 $\text{bufd}(1:iblk\text{smacro}, 1:iblk\text{sunit}) \leftarrow A(ibs:ibe, ibs2d:ibe2d)$ 、 $icps = ibe + (idst - 1) + len + 1$ 、 $icpe = isps + len - 1$ 、 $\text{bufd}(iblk\text{smacro} + 1:len + iblk\text{smacro}, 1:iblk\text{sunit}) \leftarrow A(icps:icpe, ibs2d:ibe2d)$ を演算する。このコピーは 1 次元目をスレッド数に分割して各スレッドで並列に処理する。

【0056】

ステップ S 51 では、全ノードで送受信を行う。すなわち、 bufd の内容お $idst$ 番目のノードに送り、 bufs に受信する。ステップ S 52 においては、送受信の完了を待つ。ステップ S 53 では、バリア同期を取り、ステップ S 54 において、 $wlul$ の対応位置に、 $isrs$ 番目のノードから受けたデータを格納する。すなわち、 $icp2ds = (isrs - 1) * iblk\text{sunit} + 1$ 、 $icp2de = icp2ds + iblk\text{sunit} - 1$ 、 $wlul(1:len + iblk\text{smacro}, icp2ds:$

$icp2de) \leftarrow \text{bufs}(1:len + iblk\text{sunit}, 1:iblk\text{sunit})$ を演算する。すなわち、1 次元目をスレッド数で分割して各スレッドで並列コピーする。ステップ S 55 で $iy = iy + 1$ とし、ステップ S 48 に戻る。

【0057】

図 16 及び図 17 は、サブルーチン interLU のフローである。

ステップ S 60 において、 $A(k, n / \text{numnord})$ 、 $wlul(lenblks, iblk\text{smacro})$ 、 $wlumicro(ncash)$ を引数として受ける。ここで、 $wlumicro$ を L2 キャッシュ (レベル 2 のキャッシュ) の大きさとし、各スレッドに確保されたものを受取る。 $wlul$ に LU ブロック分解する幅 $iblk\text{smacro}$ のブロックで対角ブロックとその下位ブロックを 1 次元目で numnord 個に分割した一つが各ノードの領域に格納されている。ピボットの検索と行の入れ替えに関してノード間転送を使いながら並列に LU 分解する。本サブルーチンは、再帰的に呼び出される。呼び出しが深くなる

につれてLU分解したときのブロック幅は小さくなる。このブロックをスレッド並列してLU分解したとき、各スレッドで計算する部分がキャッシュの大きさ以下になるところで、LU分解をスレッド並列化する別のサブルーチンを呼び出す。

【0058】

スレッド並列は対象となる比較的幅の小さなブロックをこのブロックの対角行列部分を各スレッドで重複して持ち、対角ブロックより下位の部分を1次元目をスレッド数で均等分割して各スレッド(CPU)にキャッシュの大きさより小さな領域wlmicroで処理できるようにコピーして処理を行う。istmicroは小さなブロックの先頭位置であり、最初1に設定される。nwidthmicroは、小さなブロックの幅であり、最初は全体のブロック幅に設定される。iblksmicromaxは、小さなブロックの最大値であり、これ以上大きいときブロック幅を更に小さく(例えば、80列に)する。nothrldはスレッド番号、numthrdはスレッド数、各ノードで重複して持つ1次元配列ip(n)に行の入れ替え情報を入れる。

【0059】

ステップS61では、 $nwidthmicro \leq iblksmicromax$ であるか否かを判断する。ステップS61の判断がYESの場合には、ステップS61において、 $iblksmicro = nwidthmicro$ とし、各ノードに分担した領域にある対角ブロックと分割したブロックが格納されているwlu(lenmacro, iblksmacro)のwlu(istmicro:lenmacro, istmicro:iblksmicro+iblksmicro-1)の部分に関して対角部分wlu(istmicro:istmicro+iblksmicro-1, istmicro:istmicro+iblksmicro-1)を対角ブロックとする。また、 $irest = istmicro + iblksmicro$ とし、wlu(irest:lenmacro, istmicro:istmicro+iblksmicro-1)を1次元目でスレッド数で均等分割したものを対角ブロックと結合して、各スレッド毎の領域wlmicroにコピーする。すなわち、 $lenmicro = (lenmacro - irest + numthrd) / numthrd$ とし、wlmicro(lenmicro+iblksmicro, iblksmicro)にコピーし、 $lenblksmicro = lenmicro + iblksmicro$ とする。そして、ステップS63で、サブルーチンLUmicroを呼び出す。これにおいては、wlmicro(lenmicro+iblksmicro, iblksmicro)を受け渡す。ステップS64では、wlmicroに分割していた部分を、対角部分は1つのスレッドから、他の部分は各スレッドのwl

umicroからwluに元々あった部分に戻す。そして、サブルーチンを抜ける。

【0060】

ステップS61の判断がNOの場合には、ステップS65において、 $nwidthmicro \geq 3 * iblksmicromax$ または、 $nwidthmicro \leq 2 * iblksmicromax$ か否かを判断する。ステップS65の判断がYESの場合には、ステップS66において、 $nwidthmicro2 = nwidthmicro / 2$ 、 $istmicro2 = istmicro + nwidthmicro2$ 、 $nwidthmicro3 = nwidthmicro - nwidthmicro2$ とし、ステップS68に進む。ステップS65の判断がNOの場合には、ステップS67において、 $nwidthmicro2 = nwidthmicro / 3$ 、 $istmicro2 = istmicro + nwidthmicro2$ 、 $nwidthmicro3 = nwidthmicro - nwidthmicro2$ とし、ステップS68に進む。ステップS68では、 $istmicro$ は、そのまま、 $nwidthmicro$ として $nwidthmicro2$ を渡してサブルーチンinterLUを呼び出す。

【0061】

ステップS69においては、 $wlu(istmicro:istmacro + nwidthmicro - 1)$ の部分を更新する。これは、一つのスレッドで更新すれば充分である。これに $wlu(istmicro:istmacro + nwidthmicro2 - 1, istmicro:istmacro + nwidthmicro2 - 1)$ の下三角行列の逆行列を左から乗じたもので更新する。ステップS70においては、 $wlu(istmicro2:lenmacro, istmicro2:istmicro2 + nwidthmicro3 - 1)$ を $wlu(istmicro2:lenmacro, istmicro:istmicro2 - 1) \times wlu(istmacro:istmacro + nwidthmicro2 - 1, istmacro + nwidthmicro2:istmacro + nwidthmicro - 1)$ を引いて更新する。このとき、1次元目をスレッド数で均等に分割して並列計算する。ステップS71においては、 $istmicro$ として、 $istmicro2$ 、 $nwidthmicro$ として $nwidthmicro3$ を渡してサブルーチンinterLUを呼び出し、サブルーチンを終了する。

【0062】

図18及び図19は、サブルーチンLUmicroのフローである。

ステップS75において、 $A(k, n / numnord)$ 、 $wlu1(lenblks, iblksmacro)$ 、 $wlumicro(leniblksmicro, iblksmicro)$ を引数として受ける。ここで、 $wlumicro$ をL2キャッシュの大きさの各スレッドに確保されたものを受取る。本ルーチンで $wlumicro$ に格納された部分のLU分解を行う。 ist は、LU分解するブロックの

先頭位置で最初は、1とされる。nwidthは、ブロック幅であり、最初は全体のブロック幅である。iblkmaxは、ブロック最大値（8程度）であり、これ以上小さくしない。wlumicroはスレッド毎に引数として渡される。

【0063】

ステップS76においては、 $nwidth \leq iblkmax$ か否かを判断する。ステップS76の判断がNOの時は、ステップS88に進む。ステップS76の判断がYESの場合には、ステップS77において、 $i = ist$ として、ステップS78において、 $i < ist + nwidth$ か否かを判断する。ステップS78の判断がNOの場合には、サブルーチンを抜ける。ステップS78の判断がYESの場合には、ステップS79において、各スレッドでi列目の絶対値最大の要素を見つけ、共用メモリ領域にスレッド番号順に格納する。ステップS80においては、各ノードでのノード内の最大ピボットをこの中から見つけ、この要素とノード番号、位置をセットとして全ノードが各セットを持つように通信し、各ノードで全ノードでの最大ピボットを決定する。なお、各ノードで同じ方法で最大ピボットを決定する。

【0064】

ステップS81においては、このピボット位置が各ノードが持つ対角ブロックの中か判定する。ステップS81の判断がNOの場合には、ステップS85に進む。ステップS81の判断がYESの場合には、ステップS82において、最大ピボットの位置が各スレッドが重複して持つ対角ブロックの中かを判定する、ステップS82の判断がYESの場合には、ステップS83において、全ノードで保持する対角ブロック内での入れ替えで、かつ、全スレッドで重複して持つ対角部分内での入れ替えなので、スレッドで独立してピボットの入れ替えを行う。入れ替えた位置を配列ipに格納し、ステップS86に進む。ステップS82における判断がNOの場合には、ステップS84において、各ノードで独立にピボットとを交換する。交換すべきピボット行を共用域に格納して、各スレッドの対角ブロック部分と入れ替える。入れ替えた位置を配列ipに格納し、ステップS86に進む。

【0065】

ステップS85では、ノード間で通信して最大ピボットを有するノードから交

換すべき行ベクトルをコピーする。その後ピボット行を入れ替える。ステップ S 86 においては、行を更新し、ステップ S 87 において、 i 列と行で更新部分を更新し、 $i=i+1$ として、ステップ 78に戻る。

【0066】

ステップ S 88 においては、 $nwidth \geq 3 * iblksmax$ あるいは、 $nwidth \leq 2 * iblksmax$ であるか否かを判断する。ステップ S 88 の判断が YES の場合には、ステップ S 89 において、 $nwidth = nwidth / 2$ 、 $ist2 = ist + nwidth2$ とし、ステップ S 91に進む。ステップ S 88 の判断が NO の場合には、ステップ S 90 において、 $nwidth2 = nwidth / 3$ 、 $ist2 = ist + nwidth2$ 、 $nwidth3 = nwidth - nwidth2$ とし、ステップ S 91に進む。ステップ S 91 では、 ist はそのまま、 $nwidth$ として $nwidth2$ を引数として渡して、サブルーチン LUmicro を呼び出す。ステップ S 92 では、 $wlumicro(istmicro:istmacro + nwidth2 - 1, istmicro + nwidth2:istmicro + nwidthmicro - 1)$ の部分を更新する。 $wlumicro(istmicro:istmacro + nwidth2 - 1, istmicro:istmacro + nwidth2 - 1)$ の下三角行列の逆行列を左から乗したもので更新する。ステップ S 93 では、 $wlumicro(istmicro2:lenmacro, istmicro2:istmicro2 + nwidthmicro3 - 1)$ を $wlumicro(istmicro2:lenmacro, istmicro:istmicro2 - 1) \times wlumicro(istmicro:istmacro + nwidth2 - 1, ist + nwidth2:ist + nwidthmicro - 1)$ を引いて更新する。ステップ S 94 においては ist として $ist2$ 、 $nwidth$ として $nwidth3$ を引数として受け渡して、サブルーチン LUmicro を呼び出して、サブルーチンを抜ける。

【0067】

図 20 は、サブルーチン btoc のフローである。

ステップ S 100 において、 $A(k, n / numnord)$ 、 $wlul(lenblks, iblksmacro)$ 、 $bufs(lenblks, iblksunit)$ 、 $bufd(lenblks, iblksunit)$ を引数で受けて、各ノードの i 番目の幅 $iblksunit$ のブロックを $numnord$ 個束ねたものの対角ブロック行列部分 $iblksmacro \times iblksmacro$ より下の部分を $numnord$ 個に分割したものと対角ブロックを加えたものを各ノードに分散配置したものに転送を利用して配置を変える。

【0068】

ステップ S 101 では、 $nbase = (i - 1) * iblksmacro$ (i は呼び出しもとのメイン

ループの繰り返し回数)、 $ibs = nbase + 1$ 、 $ibe = nbase + iblksmacro$ 、 $len = (n - ibe) / numnord$ 、 $nbase2d = (i - 1) * iblksunit$ 、 $ibs2d = nbase2d + 1$ 、 $ibe2d = ibs2d + iblksunit$ とし、送信データ数は、 $lensend = (len + iblksmacro) * iblksunit$ とする。

【0069】

ステップS102において、 $iy = 1$ とし、ステップS103において、 $iy > numnord$ か否かを判断する。ステップS103の判断がYESの場合、サブルーチンを抜ける。ステップS103の判断がNOの場合には、ステップS104において、送信する部分、受信する部分を決める。すなわち、 $idst = \text{mod}(nonord - 1 + iy - 1, numnord) + 1$ 、 $isrs = \text{mod}(nonord - 1 + numnord - iy + 1, numnord) + 1$ とする。ステップS105においては、計算結果が格納されている $wlul$ から元の位置に配置を戻すための送信のためにバッファに格納する。 $idst$ 番目のノードに対応部分を送る。すなわち、 $icp2ds = (idst - 1) * iblksunit + 1$ 、 $icp2de = icp2ds + iblksunit - 1$ 、 $bufd(1:len + iblksunit, 1:iblksunit) \leftarrow wlul(1:len + iblksmacro, icp2ds:icp2de)$ とする。1次元目をスレッド数で分割して各スレッドで並列コピーする。

【0070】

ステップS106では、全ノードで送受信する。 $bufd$ の内容を $idst$ 番目のノードに送り、 $bufs$ に受信する。ステップS107で送受信の完了を待ち、ステップS108において、バリア同期を取る。ステップS109では、各ノードで自分に割り付いている幅 $iblksunit$ の対角ブロック部分と、その下の部分のブロックの1次元目を $numnord$ で分割した部分で再配置したときの部分（転送先のノード数番目のもの）を元々あった部分に格納する。 $A(ibs:ibe, ibs2d:ibe2d) \leftarrow bufs(1:iblksmacro, 1:iblksunit)$ 、 $icps = ibe + (isrs - 1) * len + 1$ 、 $icpe = isrs + len - 1$ 、 $A(icps:icpe, ibs2d:ibe2d) \leftarrow bufs(iblksmacro + 1:len + iblksmacro, 1:iblksunit)$ とする。このコピーは1次元目をスレッド数に分割して各スレッドで列毎に処理する。

【0071】

ステップS110においては、 $iy = iy + 1$ として、ステップS103に戻る。

図21は、サブルーチン $exrw$ のフローである。

このサブルーチンは、行の入れ替え及び行ブロックの更新を行うものである。

【0072】

ステップS115においては、 $A(k, n/\text{numnord})$ 、 $wlul(\text{lenblks}, \text{iblkmacro})$ を引数として受ける。 $wlul(1:\text{iblkmacro}, 1:\text{iblkmacro})$ には、LU分解された対角部分を全ノードが重複して持っている。 $\text{nbdiag}=(i-1)*\text{iblkmacro}$ とする。 i は呼び出し元のサブルーチン pLU のメインループの繰り返し回数である。また、ピボットの入れ替えの情報が、 $ip(\text{nbdiag}+1:\text{nbdiag}+\text{iblkmacro})$ に格納されている。

【0073】

ステップS116では、 $\text{nbase}=i*\text{iblkunit}$ (i は呼び出しもとのサブルーチン pLU のメインループの繰り返し回数)、 $\text{irows}=\text{nbase}+1$ 、 $\text{irowe}=n/\text{numnord}$ 、 $\text{len}=(\text{irowe}-\text{irows}+1)/\text{numthrd}$ 、 $\text{is}=\text{nbase}+(\text{nothrd}-1)*\text{len}+1$ 、 $\text{ie}=\min(\text{irowe}, \text{is}+\text{len}-1)$ とする。ステップS117では、 $\text{ix}=\text{is}$ とする。

【0074】

ステップS118では、 $\text{is} \leq \text{ie}$ であるか否かを判断する。ステップS118の判断がNOの場合には、ステップS125に進む。ステップS118の判断がYESの場合には、ステップS119において、 $\text{nbdiag}=(i-1)*\text{iblkmacro}$ 、 $j=\text{nbdiag}+1$ として、ステップS120において、 $j \leq \text{nbdiag}+\text{iblkmacro}$ であるか否かを判断する。ステップS120の判断がNOの場合には、ステップS124に進む。ステップS120の判断がYESの場合には、ステップS121において、 $ip(j) > j$ か否かを判断する。ステップS121の判断がNOの場合には、ステップS123に進む。ステップS121の判断がYESの場合には、ステップS122において、 $A(j, \text{ix})$ と $A(ip(j), \text{ix})$ を入れ替えて、ステップS123に進む。ステップS123においては、 $j=j+1$ として、ステップS120に戻る。

【0075】

ステップS124においては、 $\text{ix}=\text{ix}+1$ とし、ステップS118に戻る。

ステップS125においては、バリア同期 (全ノード、全スレッド) を取る。ステップS126においては、 $A(\text{nbdiag}+1:\text{nbdiag}+\text{iblkmacro}, \text{is}:\text{ie}) \leftarrow \text{TRL}(wlul(i:\text{iblkmacro}, 1:\text{iblkmacro}))^{-1} \times A(\text{nbdiag}+1:\text{nbdiag}+\text{iblkmacro}, \text{is}:\text{ie})$ を全ノード、全スレッドで更新する。ここで、 $\text{TRL}(B)$ は、行列 B の下三角

部分を示す。ステップ S 1 2 7 では、バリア同期（全ノード、全スレッド）を取って、サブルーチンを抜ける。

【0076】

図 2 2 及び図 2 3 は、サブルーチン mmcbt のフローである。

ステップ S 1 3 0 において、 $A(k, n/\text{numnord})$ 、 $wlul(\text{lenblks}, \text{iblkmacro})$ 、 $wlu2(\text{lenblks}, \text{iblkmacro})$ を引数として受ける。 $wlul$ に、ブロック幅 iblkmacro のブロックを LU 分解した結果で、対角ブロックとその下位ブロックを 1 次元目で numnord 個に分割した一つが格納されている。分割した順にノード番号に対応し、ノードに再配置される。これをノードのリングに沿って転送しながら（計算と同時に行う）行列積を行いながら更新する。計算の裏で性能に影響を与えないので計算に直接使用しない対角ブロック部分も一緒に送る。

【0077】

ステップ S 1 3 1 では、 $\text{nbase}=(i-1)*\text{iblkmacro}$ （ i は呼び出しもとのサブルーチン p LU のメインループの繰り返し回数）、 $\text{ibs}=\text{nbase}+1$ 、 $\text{ibe}=\text{nbase}+\text{iblkmacro}$ 、 $\text{len}=(\text{n}-\text{ibe})/\text{numnord}$ 、 $\text{nbase2d}=(i-1)*\text{iblkunit}$ 、 $\text{ibs2d}=\text{nbase2d}+1$ 、 $\text{ibe2d}=\text{ibs2d}+\text{iblkunit}$ 、 $\text{n2d}=\text{n}/\text{numnord}$ 、 $\text{lensend}=\text{len}+\text{iblkmacro}$ とし、送信データ数は、 $\text{nwlen}=\text{lensend}*\text{iblkmacro}$ とする。

【0078】

ステップ S 1 3 2 において、 $\text{iy}=1$ （初期値を設定）、 $\text{idst}=\text{mod}(\text{nonord}, \text{numnord})+1$ （送り先ノード番号（隣ノード））、 $\text{isrs}=\text{mod}(\text{nonord}-1+\text{numnord}-1, \text{numnord})+1$ （発信元ノード番号）、 $\text{ibp}=\text{idst}$ とする。

【0079】

ステップ S 1 3 3 において、 $\text{iy} > \text{numnord}$ であるか否かを判断する。ステップ S 1 3 3 の判断が YES の場合には、サブルーチンを抜ける。ステップ S 1 3 3 の判断が NO の場合には、ステップ S 1 3 4 において、 $\text{iy}=1$ か否かを判断する。ステップ S 1 3 4 の判断が YES の場合には、ステップ S 1 3 6 に進む。ステップ S 1 3 4 の判断が NO の場合には、ステップ S 1 3 5 において送受信の官僚を待つ。ステップ S 1 3 6 では、 $\text{iy}=\text{numnord}$ （奇数の最後）であるか否かを判断する。ステップ S 1 3 6 の判断が YES の場合には、ステップ S 1 3 8 に進む。ステ

ップS 136の判断がNOの場合には、ステップS 137において、送受信を行う。wlulの内容を（対角ブロックも含めて）隣のノード（ノード番号idst）に送る。かつ、wlu2に（ノード番号isrsから）送られてくるデータを格納する。送受信データ長はnwlenとする。

【0080】

ステップS 138において、wlulのデータを使った更新のポジションを計算する。 $ibp = \text{mod}(ibp-1 + \text{numnord}-1, \text{numnord}) + 1$ 、 $ncptr = nbe + (ibp-1) * len + 1$ （1次元目の開始位置）とする。ステップS 139では、行列積を計算するサブルーチンpmmを呼び出す。このときwlulを引き渡す。ステップS 140において、 $iy = \text{numnord}$ （最後の処理が終わった）か否かを判断する。ステップS 140の判断がYESの場合には、サブルーチンを抜ける。ステップS 140の判断がNOの場合には、ステップS 141において、行列積演算と同時に行っている送受信の完了を待つ。ステップS 142において、 $iy = \text{numnord}-1$ （偶数の最後）であるか否かを判断する。ステップS 142の判断がNOの場合には、ステップS 144に進む。ステップS 142の判断がNOの場合には、ステップS 143において、送受信を行う。すなわち、wlu2の内容を（対角ブロックも含めて）隣のノード（ノード番号idst）に送る。かつ、wlulに（ノード番号isrsから）送られてくるデータを格納する。送受信データ長はnwlenとする。

【0081】

ステップS 144では、wlu2のデータを使った更新のポジションを計算する。すなわち、 $ibp = \text{mod}(ibp-1 + \text{numnord}-1, \text{numnord}) + 1$ 、 $ncptr = nbe + (ibp-1) * len + 1$ （1次元目の開始位置）とする。

【0082】

ステップS 145では、行列積を計算するサブルーチンpmmを呼び出す。このとき、wlu2を引き渡す。ステップS 146において、 $iy = iy + 2$ と、2を加えて、ステップS 133に戻る。

【0083】

図24は、サブルーチンpmmのフローである。

ステップS 150において、 $A(k, n / \text{numnord})$ 、wlul (lenblks, iblksm

acro)、もしくは、wlu2(lenblks, iblksmacro) を wlu(lenblks, iblksmacro) に受ける。呼び出し元から渡された1次元目の開始位置ncptrを使って正方形の領域を更新する。is2d=i*iblkunit+1、ie2d=n/numnord、len=ie2d-is2d+1、isld=ncptr、ield=nptr+len-1 (iはサブルーチンpLUの繰り返し数)、A(isld:ield, is2d:ie2d) = A(isld:ield, is2d:ie2d) - wlu(iblksmacro+1:iblksmacro+len, 1:iblksmacro) × A(isld-iblksmacro:isld-1, is2d:ie2d) (式1) とする。

【0084】

ステップS151において、並列に処理するスレッド数の平方根を求めて切り上げる。numroot=int(sqrt(numthrd))、もし、sqrt(numthrd)-numrootが0でないなら、numroot=numroot+1とする。ここで、intは小数点以下切り捨て、sqrtは、平方根である。ステップS152において、m1=numroot、m2=numroot、mx=m1とする。ステップS153において、m1=mx、mx=mx-1、mm=mx×m2とする。ステップS154において、mm<numthrdであるか否かを判断する。ステップS154の判断がNOの場合には、ステップS153に戻る。ステップS154の判断がYESの場合には、ステップS155において、更新する領域を1次元目をm1等分する。2次元目をm2等分して、m1×m2個の矩形にする。そのうち、numthrd個を各スレッドに割り当てて、(式1)の対応部分を並列に計算する。(1, 1)、(1, 2)、・・・(1, m2)、(2, 1)・・・と2次元目の方向に順番にスレッドを対応付けていく。

【0085】

ステップS156において、m1*m2-numthrd>0か否かを判断する。ステップS156の判断がYESの場合には、ステップS158に進む。ステップS156の判断がNOの場合には、ステップS157において、残りの矩形は最後の矩形の最後の行、1行の最後からm1*m2-numthrd個が更新されずに残っている。この矩形を結合して1つの矩形と考え、2次元目をスレッド数numthrdで分割して(式1)の対応部分を並列に計算する。そして、ステップS158において、バリア同期(スレッド間)をとって、サブルーチンを抜ける。

【0086】

図25は、サブルーチンfbluのフローである。

ステップ S 1 6 0 において、 $A(k, n/\text{numnord})$ 、 $wlul(\text{iblkmacro}, \text{iblkmacro})$ 、 $bufs(\text{iblkmacro}, \text{iblkunit})$ 、 $bufd(\text{iblkmacro}, \text{iblkunit})$ を引数で受けて、各ノードの幅 iblkunit の最後のブロックを numnord 個束ねたものを各ノードで重複して持つように利用不足部分を各ノードに送る。各ノードが $\text{iblkmacro} \times \text{iblkmacro}$ のブロックを重複して持った後、各ノードで同じ行列に対して LU 分解を行う。LU 分解が完了したら、各ノードに配置されていた部分をコピーバックする。

【0087】

ステップ S 1 6 1 では、 $\text{nbase} = n - \text{iblkmacro}$ 、 $\text{ibs} = \text{nbase} + 1$ 、 $\text{ibe} = n$ 、 $\text{len} = \text{iblkmacro}$ 、 $\text{nbase2d} = (i-1) \times \text{iblkunit}$ 、 $\text{ibs2d} = n/\text{numnord} - \text{iblkunit} + 1$ 、 $\text{ibe2d} = n/\text{numnord}$ とし、送信データ数は $\text{lensend} = \text{iblkmacro} \times \text{iblkunit}$ とし、 $\text{iy} = 1$ とする。

【0088】

ステップ S 1 6 2 においては、バッファへのコピーを行う。すなわち、 $\text{bufd}(1:\text{iblkmacro}, 1:\text{iblkunit}) \leftarrow A(\text{ibs}:\text{ibe}, \text{ibs2d}:\text{ibe2d})$ とする。ステップ S 1 6 3 においては、 $\text{iy} > \text{numnord}$ か否かを判断する。ステップ S 1 6 3 の判断が YES の場合には、ステップ S 1 7 0 に進む。ステップ S 1 6 3 の判断が NO の場合には、ステップ S 1 6 4 において、送信する部分、受信する部分を決定する。すなわち、 $\text{idst} = \text{mod}(\text{nonord} - 1 + \text{iy} - 1, \text{numnord}) + 1$ 、 $\text{isrs} = \text{mod}(\text{nonord} - 1 + \text{numnord} - \text{iy} + 1, \text{numnord}) + 1$ とする。ステップ S 1 6 5 では、全ノードで送受信する。 bufd の内容を idst 番目のノードに送る。ステップ S 1 6 6 においては、 bufs にデータを受信し、送受信の完了を待つ。ステップ S 1 6 7 において、バリア同期を取り、ステップ S 1 6 8 において、 $wlul$ の対応位置に isrs 番目のノードから来たデータを格納する。 $\text{icp2ds} = (\text{isrs} - 1) \times \text{iblkunit} + 1$ 、 $\text{icp2de} = \text{icp2ds} + \text{iblkunit} - 1$ 、 $wlu(1:\text{iblkmacro}, \text{icp2ds}:\text{icp2de}) \leftarrow \text{bufs}(1:\text{iblkunit}, 1:\text{iblkunit})$ とする。ステップ S 1 6 9 において、 $\text{iy} = \text{iy} + 1$ とし、ステップ S 1 6 3 に戻る。

【0089】

ステップ S 1 7 0 では、バリア同期をとり、ステップ S 1 7 1 では、 $wlul$ の上で $\text{iblkmacro} \times \text{iblkmacro}$ の LU 分解を各ノードで重複して行う。行交換の情報は、 ip に格納する。LU 分解が終了したら、自ノード分を最後のブロックにコピ

ーバックする。すなわち、 $is=(nonord-1)*iblkunit+1$ 、 $ie=is+iblkunit-1$ 、 $A(ibs:ibe, ibs2d:ibe2d) \leftarrow wlul(1:iblkmacro, is:ie)$ として、サブルーチンを抜ける。

(付記1) 複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの1巻き分を、該1巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第1の配置ステップと、

該1巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを1次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第2の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にLU分解するLU分解ステップと、

LU分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法を情報装置に実現させるプログラム。

【0090】

(付記2) 前記LU分解は、再帰的手続きにより、各ノードの各プロセッサで並列的に行われることを特徴とする付記1に記載のプログラム。

(付記3) 前記更新ステップにおいては、各ノードが、列ブロックを計算している間に、計算し終わった部分のデータであって、他のブロックの更新に必要なデータを該計算と平行して他のノードに転送することを特徴とする付記1に記載のプログラム。

【0091】

(付記4) 前記並列計算機は、SMP (SymmetricMultiProcessor) を各ノードとするSMPノード分散メモリ型並列計算機であることを特徴とする付記1

に記載のプログラム。

【0092】

(付記5) 複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理装置であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの1巻き分を、該1巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第1の配置手段と、

該1巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離手段と、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを1次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第2の配置手段と、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にLU分解するLU分解手段と、

LU分解されたブロックを用いて、行列の他のブロックを更新する更新手段と、
を備えることを特徴とする行列処理装置。

【0093】

(付記6) 複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの1巻き分を、該1巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第1の配置ステップと、

該1巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを1次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第2の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノード

ドで並列にLU分解するLU分解ステップと、

LU分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法。

【0094】

(付記7) 複数のプロセッサとメモリを含む複数のノードをネットワークで接続した並列計算機における行列処理方法であって、

ノード毎にサイクリックに割り付けられた行列の部分の列ブロックの1巻き分を、該1巻き分をまとめたものを対象にして処理するために、各ノードに一つずつ分散して配置する第1の配置ステップと、

該1巻き分を結合したブロックに対して対角部分と該対角ブロックの下側にある列ブロックと他のブロックに分離する分離ステップと、

該対角ブロックを各ノードに冗長に配置すると共に、該列ブロックを1次元目で分割することによって得られるブロックを該複数のノードに、共に並列通信して一つずつ配置する第2の配置ステップと、

該対角ブロックと配置されたブロックを、各ノード間で通信しながら、各ノードで並列にLU分解するLU分解ステップと、

LU分解されたブロックを用いて、行列の他のブロックを更新する更新ステップと、

を備えることを特徴とする行列処理方法を情報装置に実現させるプログラムを格納する、情報装置読み取り可能な記録媒体。

【0095】

【発明の効果】

ブロックを動的に1次元目の分割にして処理し、分解した後の各ノードの情報を使って更新し、転送は計算と同時に行える。このため更新部分は負荷はノード間で完全に均等になり、転送量はノード数分の1に削減できる。

【0096】

ブロック幅を大きくすると負荷のバランスが崩れる従来の方法に対し負荷が均

等になるため並列化効率が 1 0 % 程度向上する。また、転送量が減ることによって 3 % 程度の並列化率の向上に寄与でき、転送スピードが SMP ノードの計算性能に比べて遅くなっても影響は受けにくい。

【 0 0 9 7 】

ブロック部分の LU 分解をノード間で並列計算することによって、ブロック幅を大きくしたとき並列化出来ない部分の割合が増加するため並列化効率が落ちる部分をキャンセルできて約 1 0 % の並列化効率の向上が見込める。また、ブロック LU 分解を、ミクロなブロックをベースにした再帰的プログラミングを使うことで、対角ブロックも含めて SMP の並列化ができて SMP での並列処理での性能劣化を抑えることができる。

【図面の簡単な説明】

【図 1】

本発明の実施形態が適用される SMP ノード分散メモリ型並列計算機の概略全体構成を示す図である。

【図 2】

本発明の実施形態に従った全体の処理フローチャートである。

【図 3】

本発明の実施形態の一般概念図である。

【図 4】

比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図（その 1）である。

【図 5】

比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図（その 2）である。

【図 6】

図 4 及び図 5 で配置されたブロックの更新処理を説明する図である。

【図 7】

再帰的な LU 分解の手順を説明する図である。

【図 8】

対角部分以外の部分ブロックの更新について説明する図である。

【図 9】

行ブロックの更新処理を説明する図（その 1）である。

【図 1 0】

行ブロックの更新処理を説明する図（その 2）である。

【図 1 1】

行ブロックの更新処理を説明する図（その 3）である。

【図 1 2】

本発明の実施形態のフローチャート（その 1）である。

【図 1 3】

本発明の実施形態のフローチャート（その 2）である。

【図 1 4】

本発明の実施形態のフローチャート（その 3）である。

【図 1 5】

本発明の実施形態のフローチャート（その 4）である。

【図 1 6】

本発明の実施形態のフローチャート（その 5）である。

【図 1 7】

本発明の実施形態のフローチャート（その 6）である。

【図 1 8】

本発明の実施形態のフローチャート（その 7）である。

【図 1 9】

本発明の実施形態のフローチャート（その 8）である。

【図 2 0】

本発明の実施形態のフローチャート（その 9）である。

【図 2 1】

本発明の実施形態のフローチャート（その 1 0）である。

【図 2 2】

本発明の実施形態のフローチャート（その 1 1）である。

【図 2 3】

本発明の実施形態のフローチャート（その 1 2）である。

【図 2 4】

本発明の実施形態のフローチャート（その 1 3）である。

【図 2 5】

本発明の実施形態のフローチャート（その 1 4）である。

【図 2 6】

スーパスカラ並列計算機用 L U 分解法のアルゴリズムを概略説明する図である。

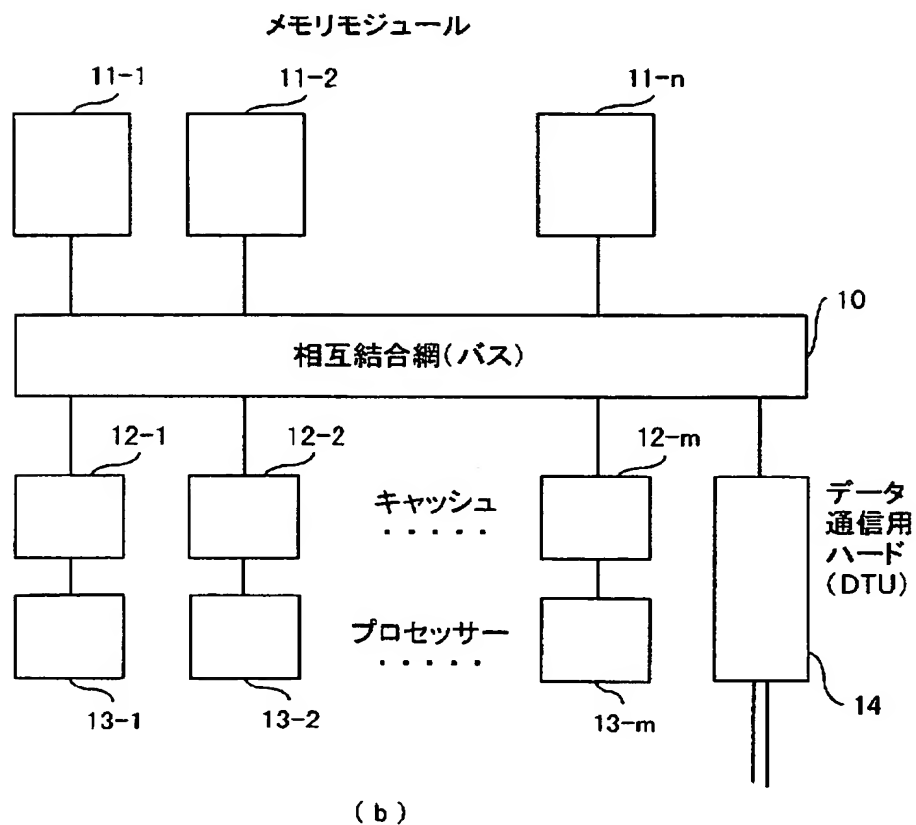
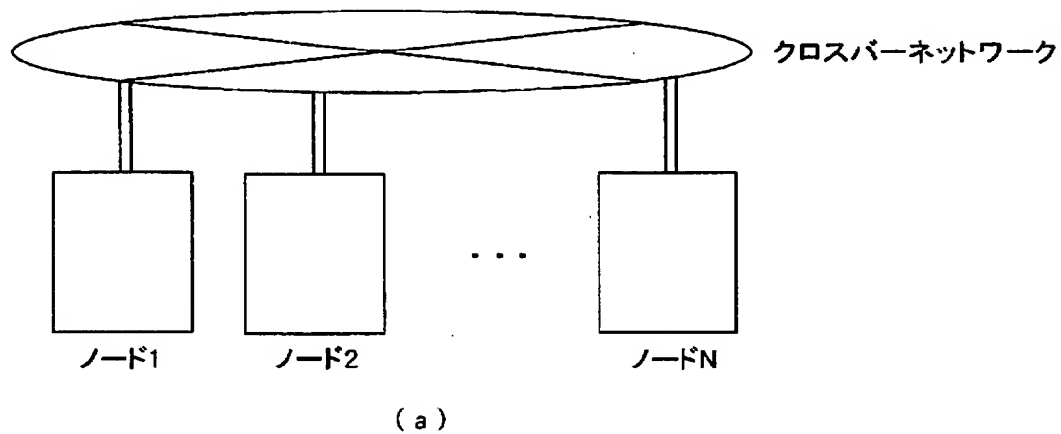
【符号の説明】

- 1 0 相互結合網（バス）
- 1 1 - 1 ~ 1 1 - n メモリモジュール
- 1 2 - 1 ~ 1 2 - m キャッシュ
- 1 3 - 1 ~ 1 3 - m プロセッサ
- 1 4 データ通信用ハード（D T U）

【書類名】 図面

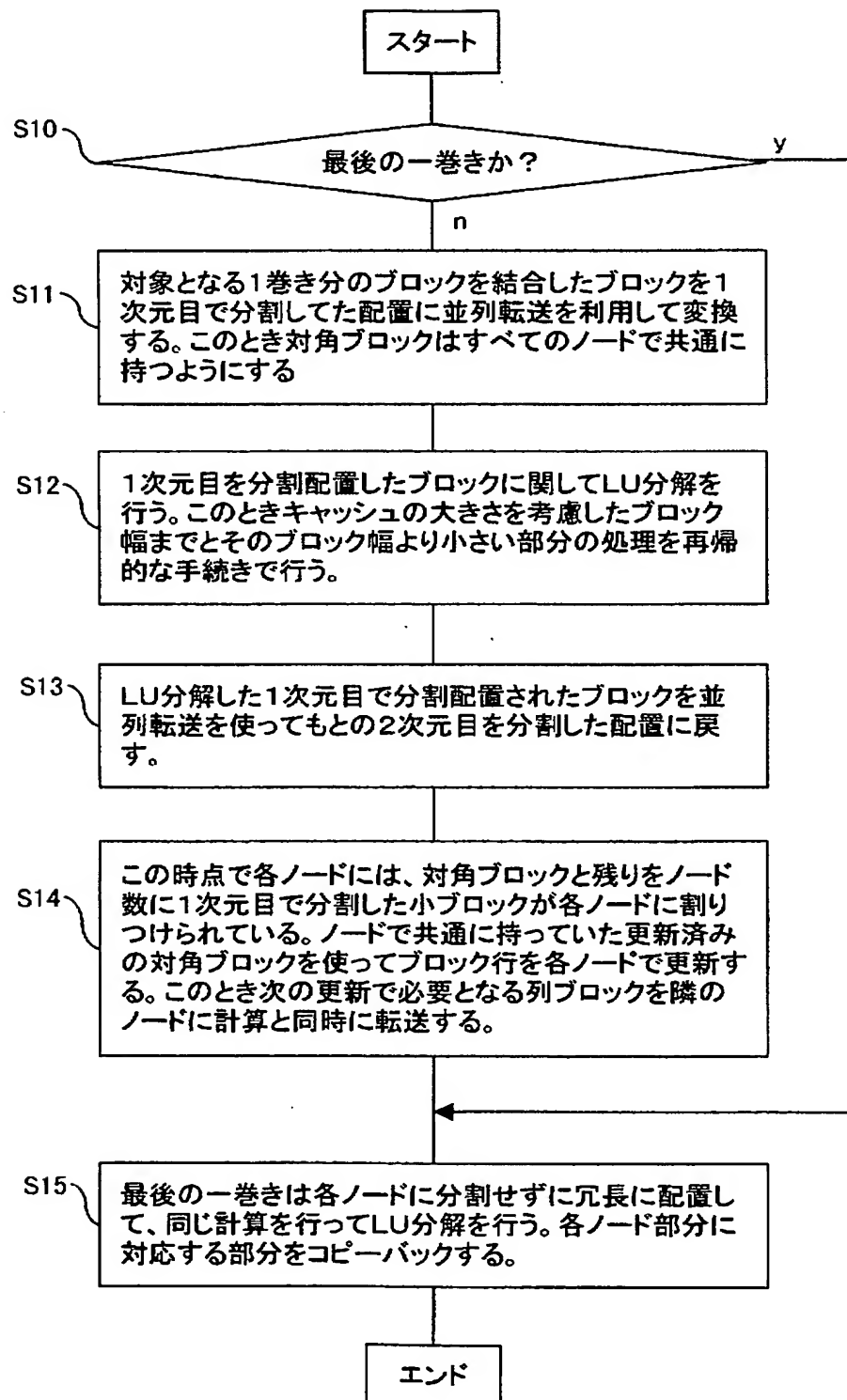
【図 1】

本発明の実施形態が適用されるSMPノード分散
メモリ型並列計算機の概略全体構成を示す図



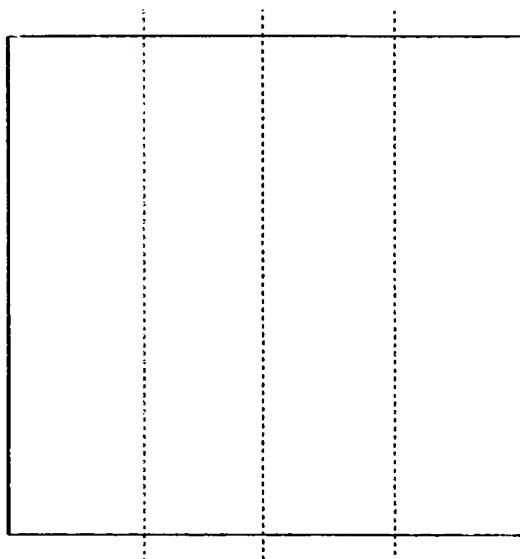
【図 2】

本発明の実施形態に従った全体の処理フローチャート



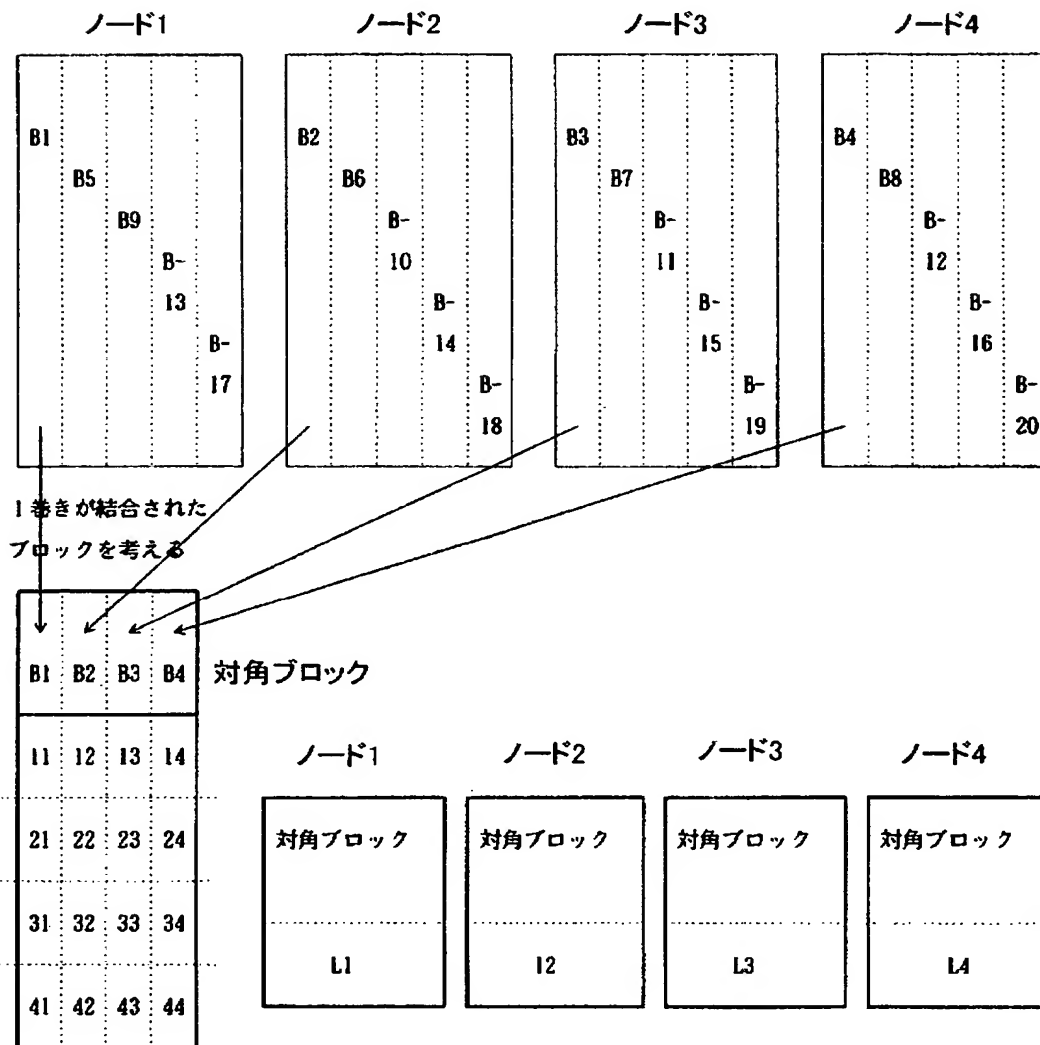
【図 3】

本発明の実施形態の一般概念図



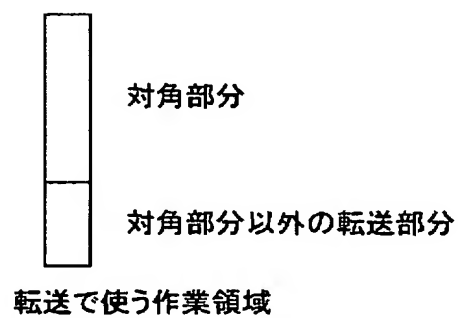
【図 4】

比較的ブロック幅の小さなブロックをサイクリックに配置した状態を説明する図(その1)



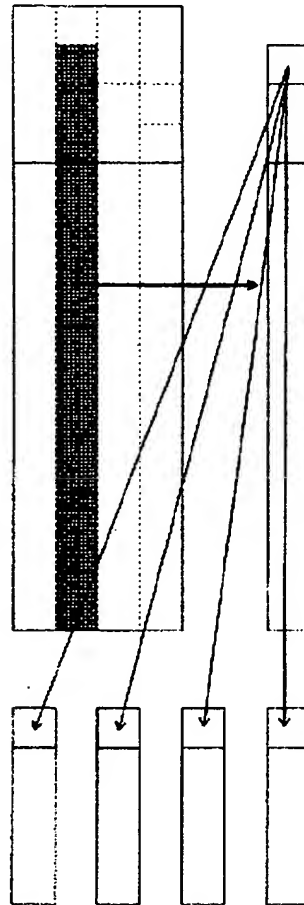
【図 5】

比較的ブロック幅の小さなブロックをサイクリックに
配置した状態を説明する図(その2)



【図 6】

図4及び図5で配置されたブロックの更新処理を説明する図

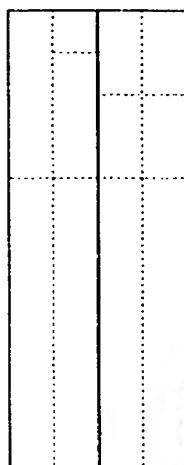


共用作業域

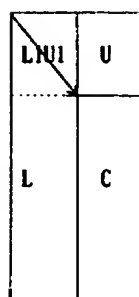


【図 7】

再帰的なLU分解の手順を説明する図



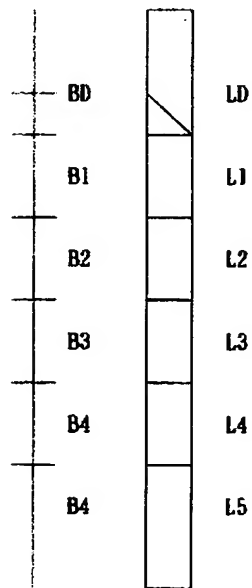
(a)



(b)

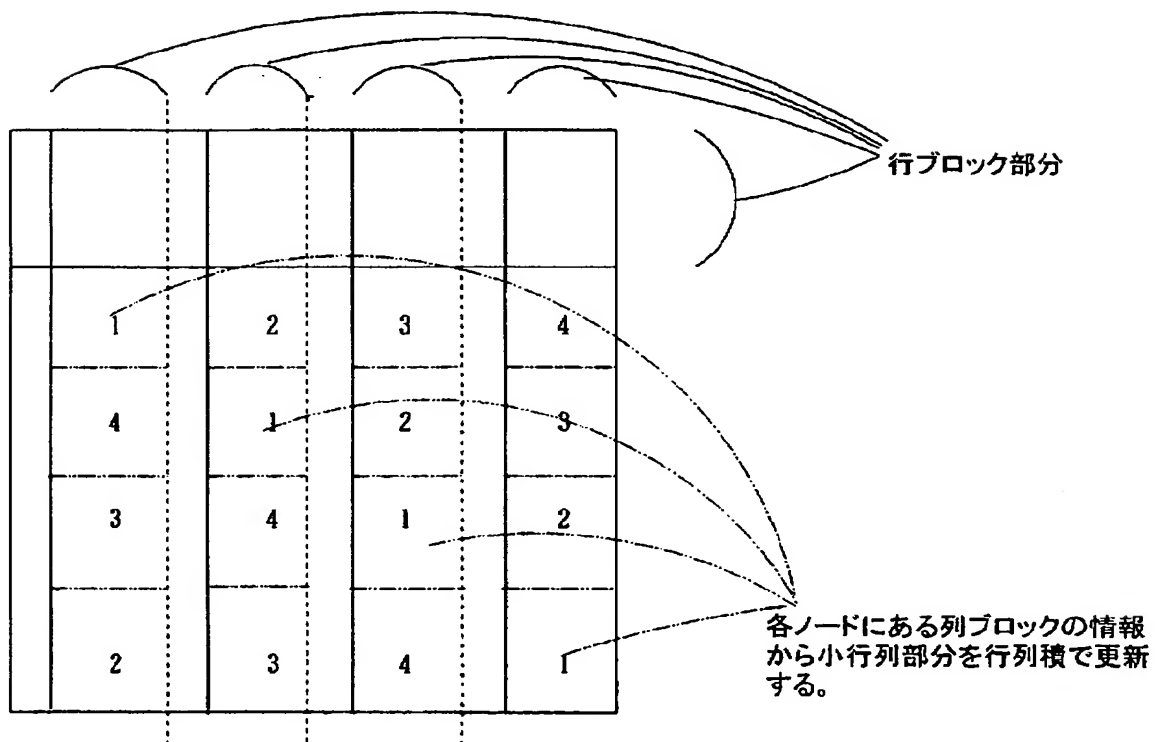
【図 8】

対角部分以外の部分ブロックの更新について説明する図



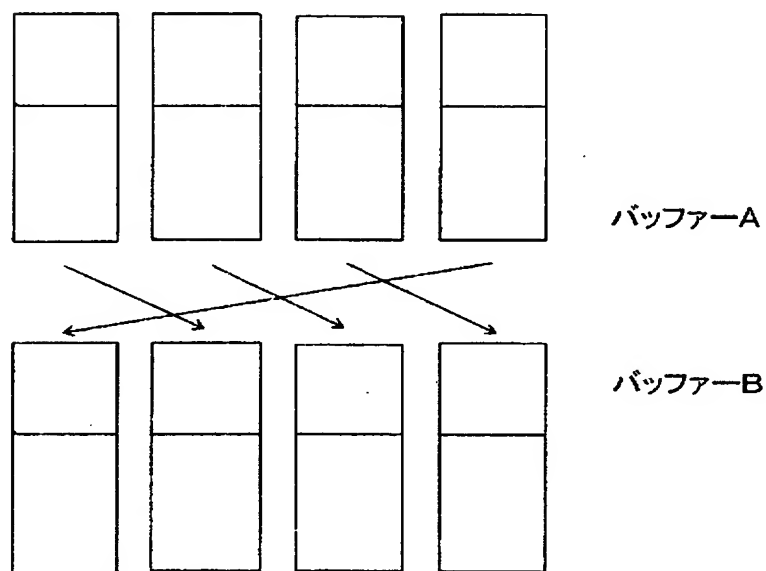
【図 9】

行ブロックの更新処理を説明する図(その1)



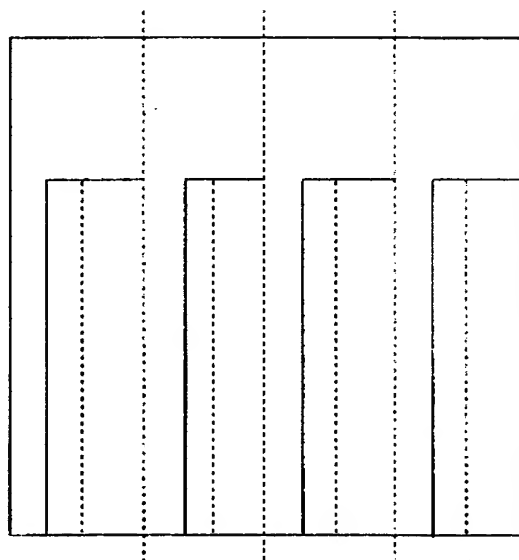
【図 10】

行ブロックの更新処理を説明する図(その2)



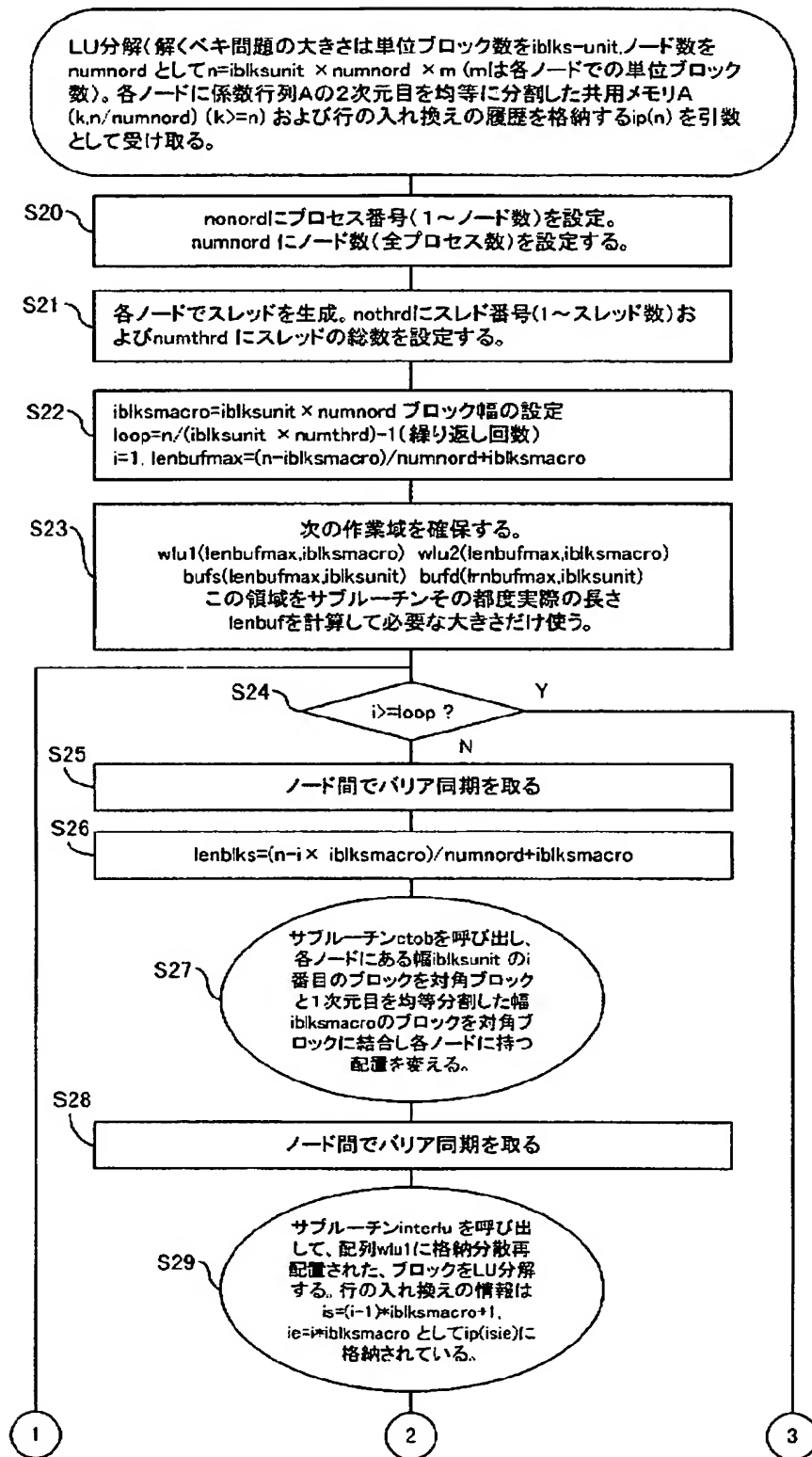
【図 11】

行ブロックの更新処理を説明する図(その3)



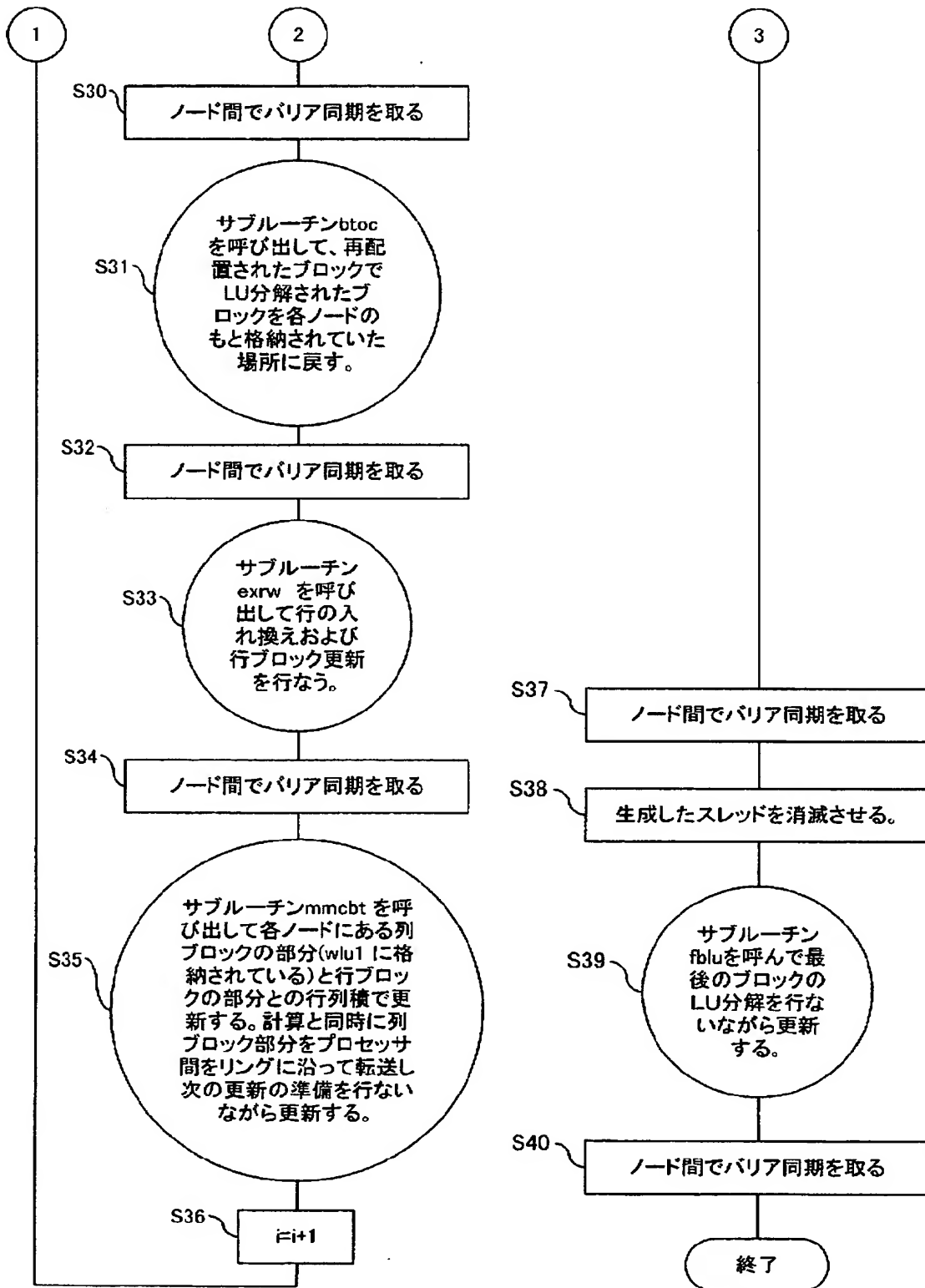
【図 12】

本発明の実施形態のフローチャート(その1)



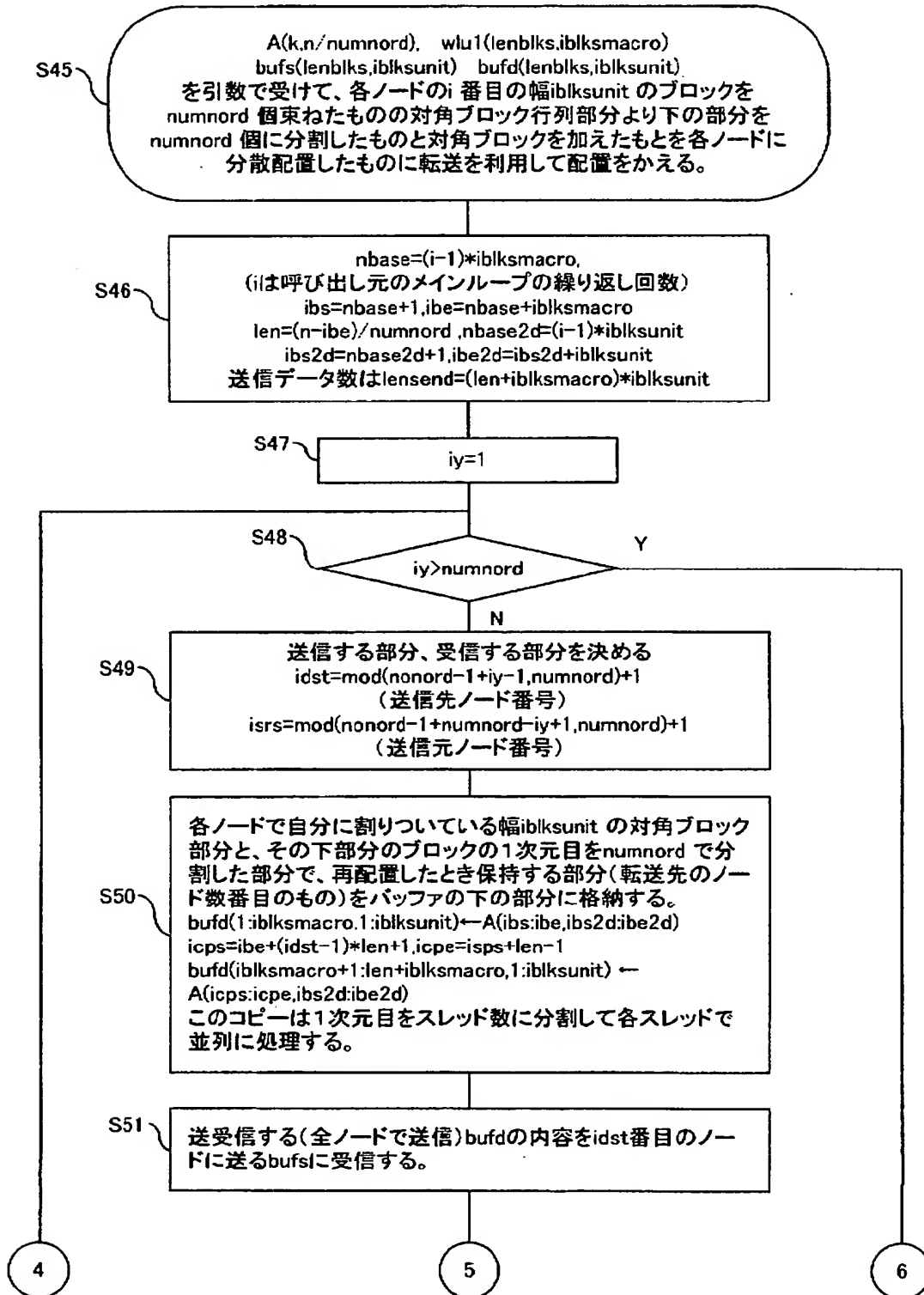
【図 13】

本発明の実施形態のフローチャート(その2)



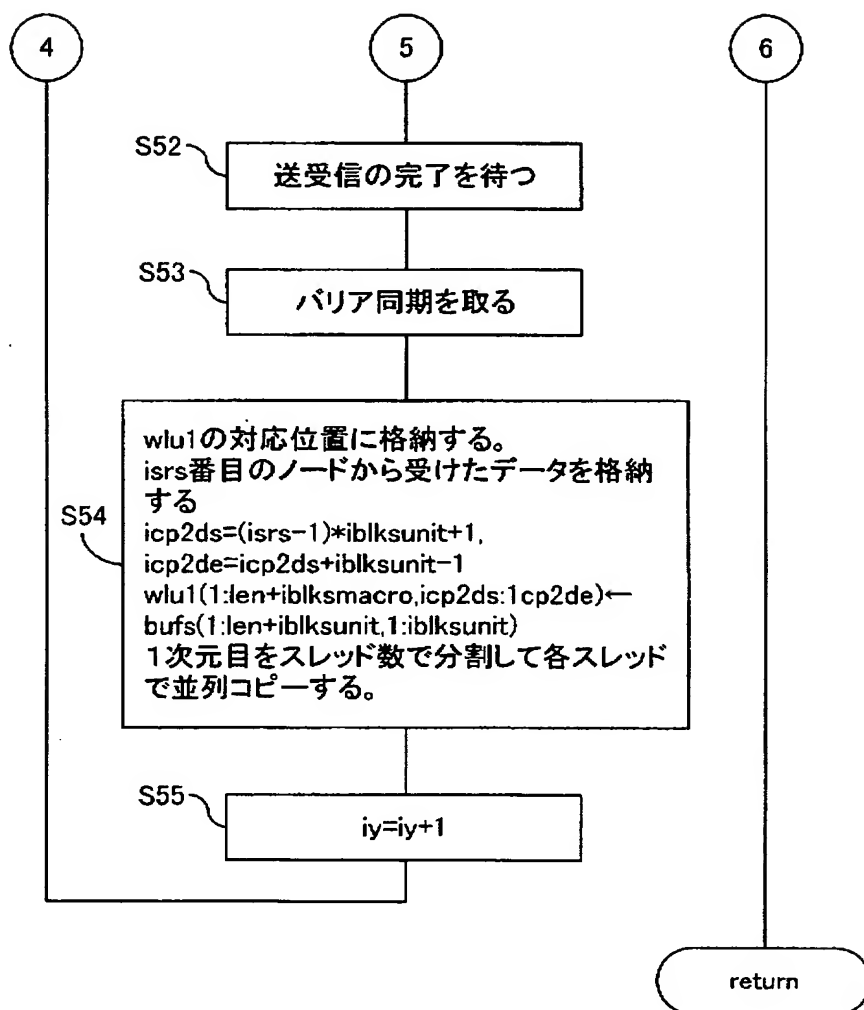
【図 14】

本発明の実施形態のフローチャート(その3)



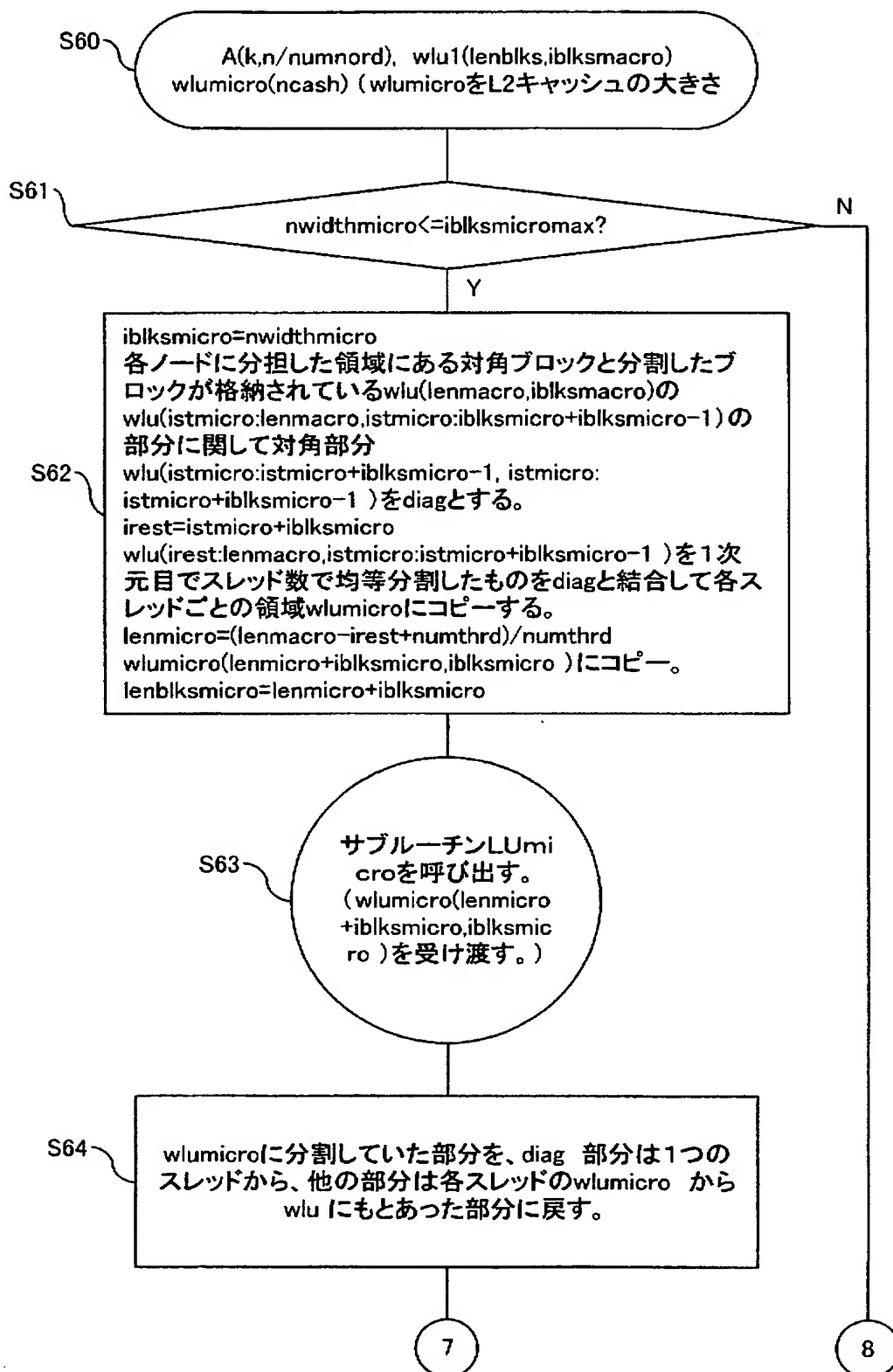
【図 15】

本発明の実施形態のフローチャート(その4)



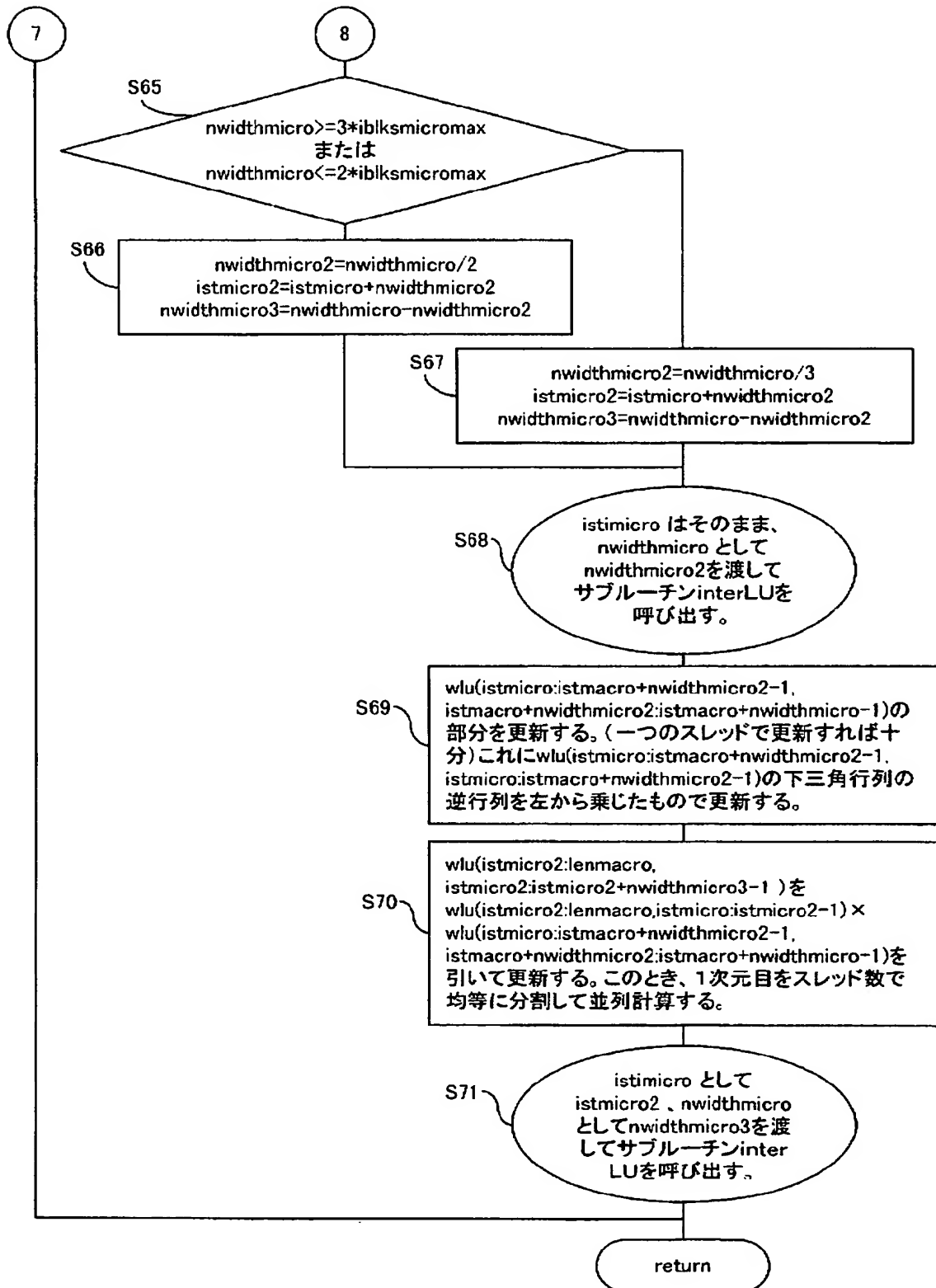
【図 16】

本発明の実施形態のフローチャート(その5)



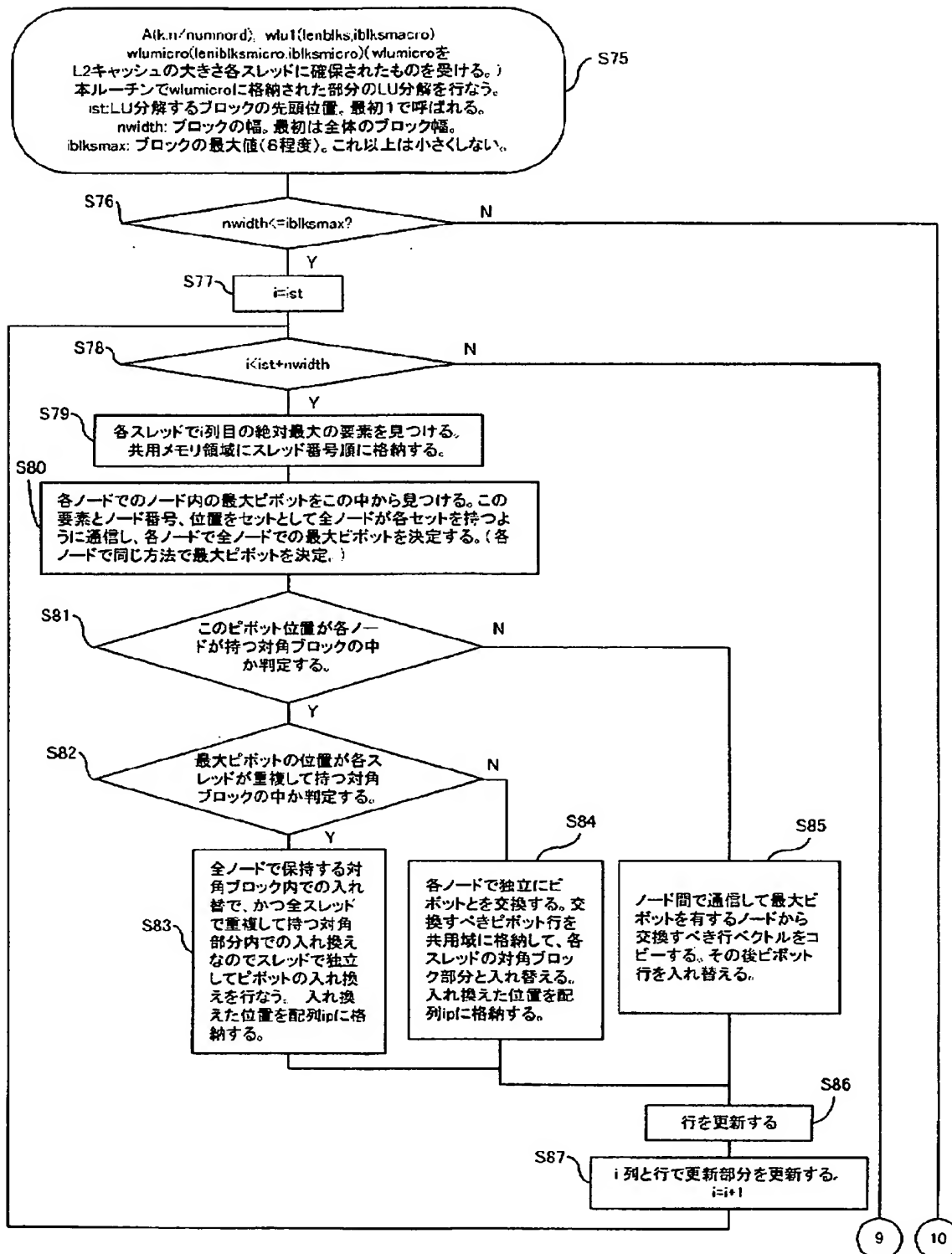
【図 17】

本発明の実施形態のフローチャート(その6)



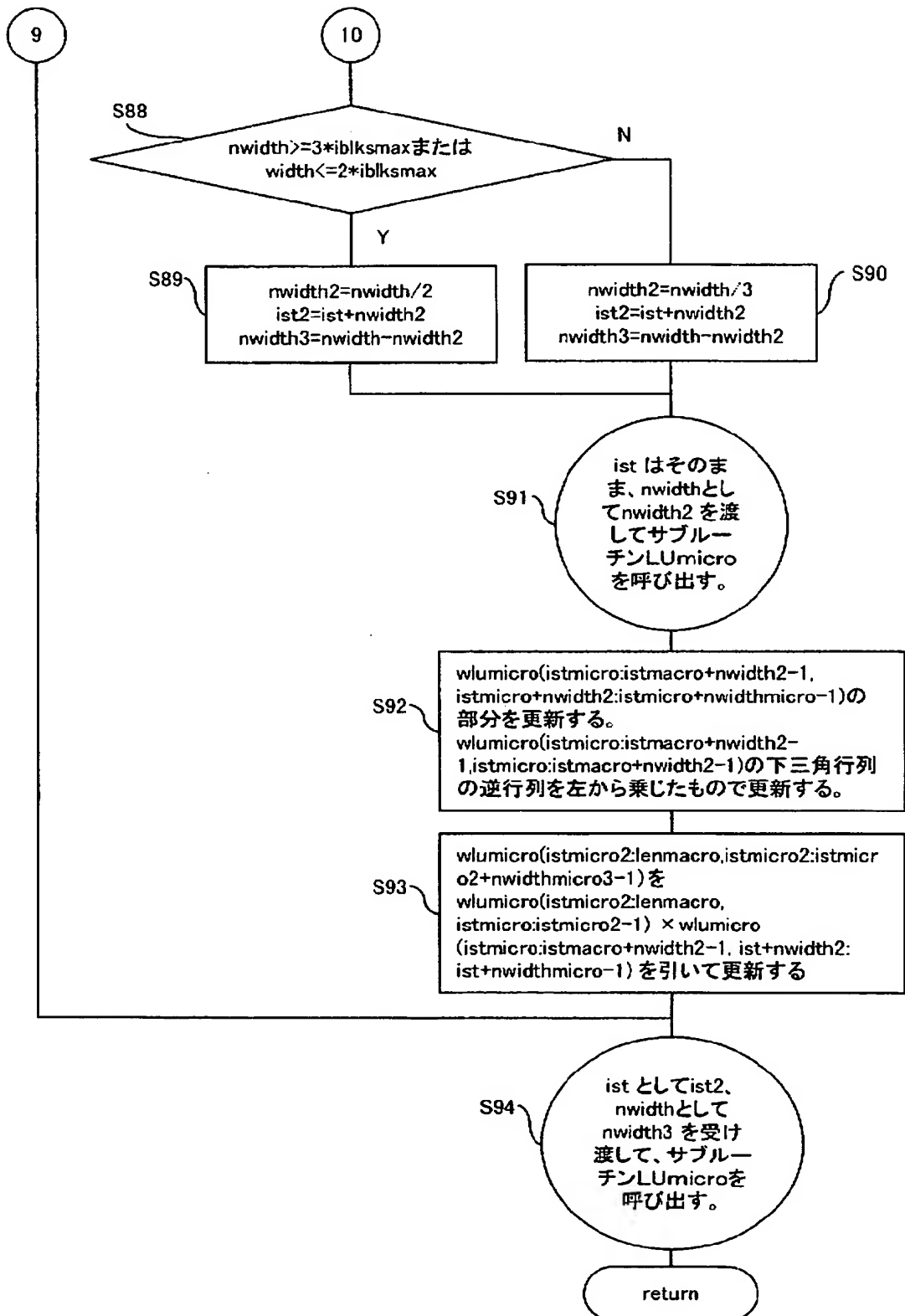
【図 18】

本発明の実施形態のフローチャート(その7)



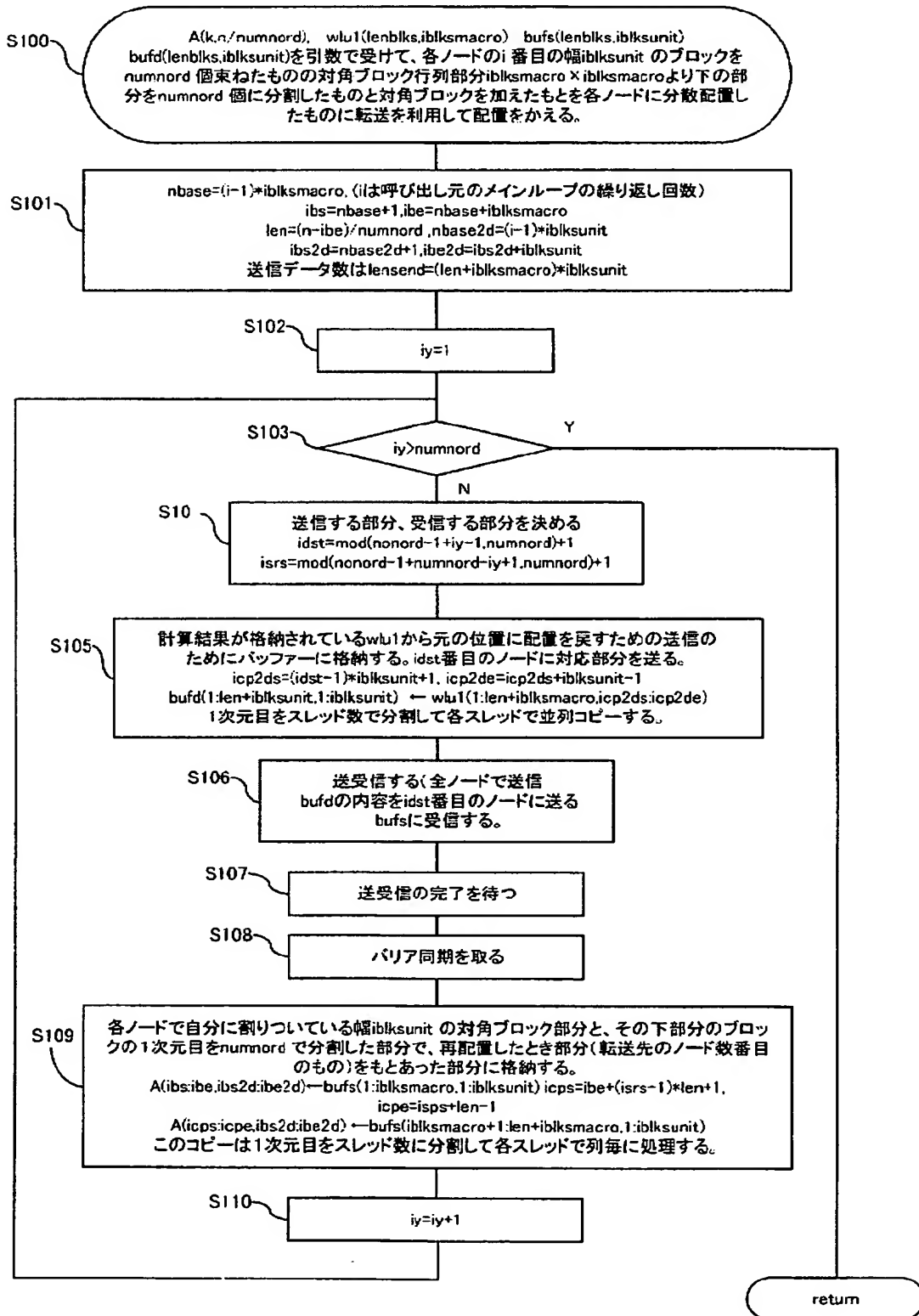
【図 19】

本発明の実施形態のフローチャート(その8)



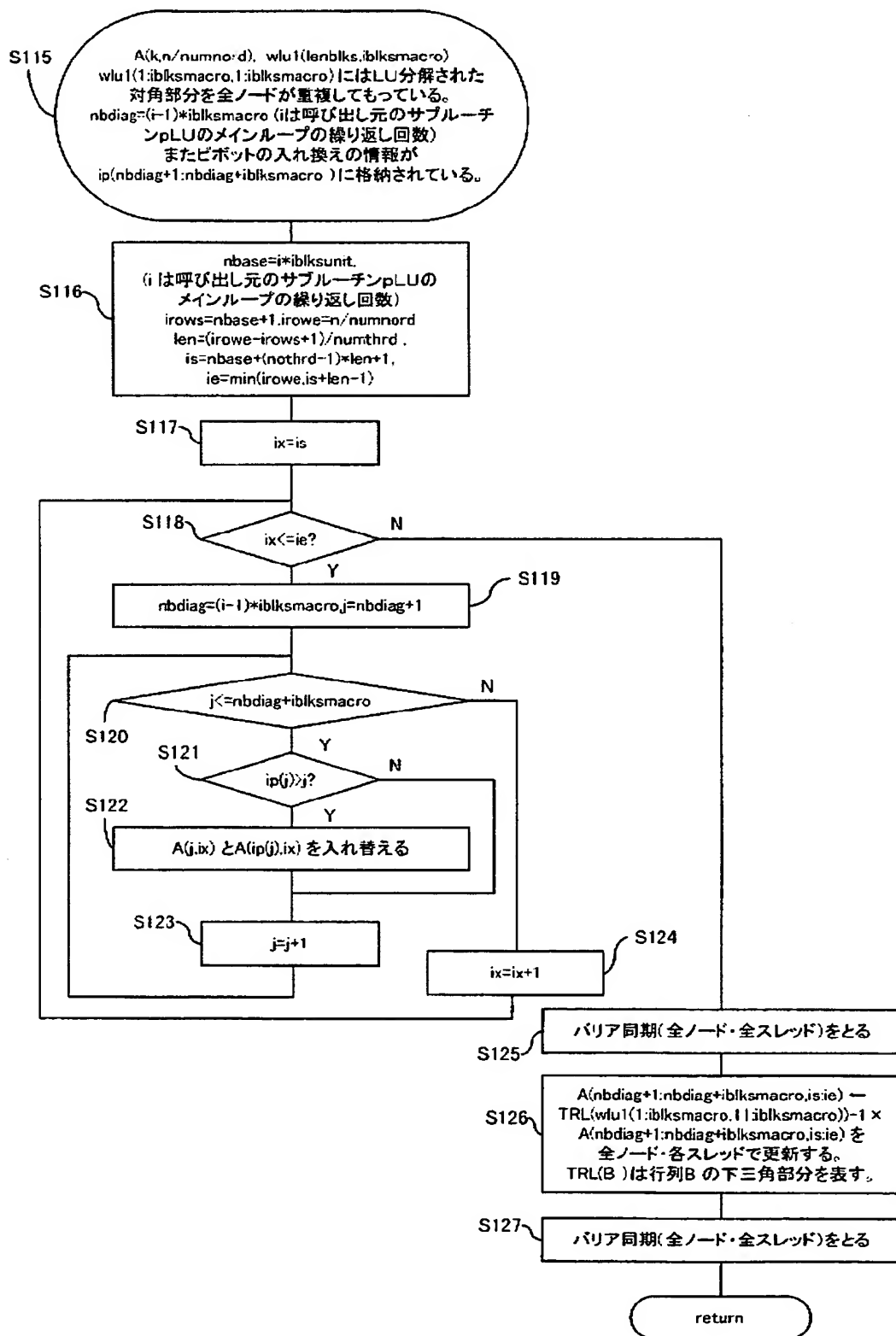
【図 20】

本発明の実施形態のフローチャート(その9)



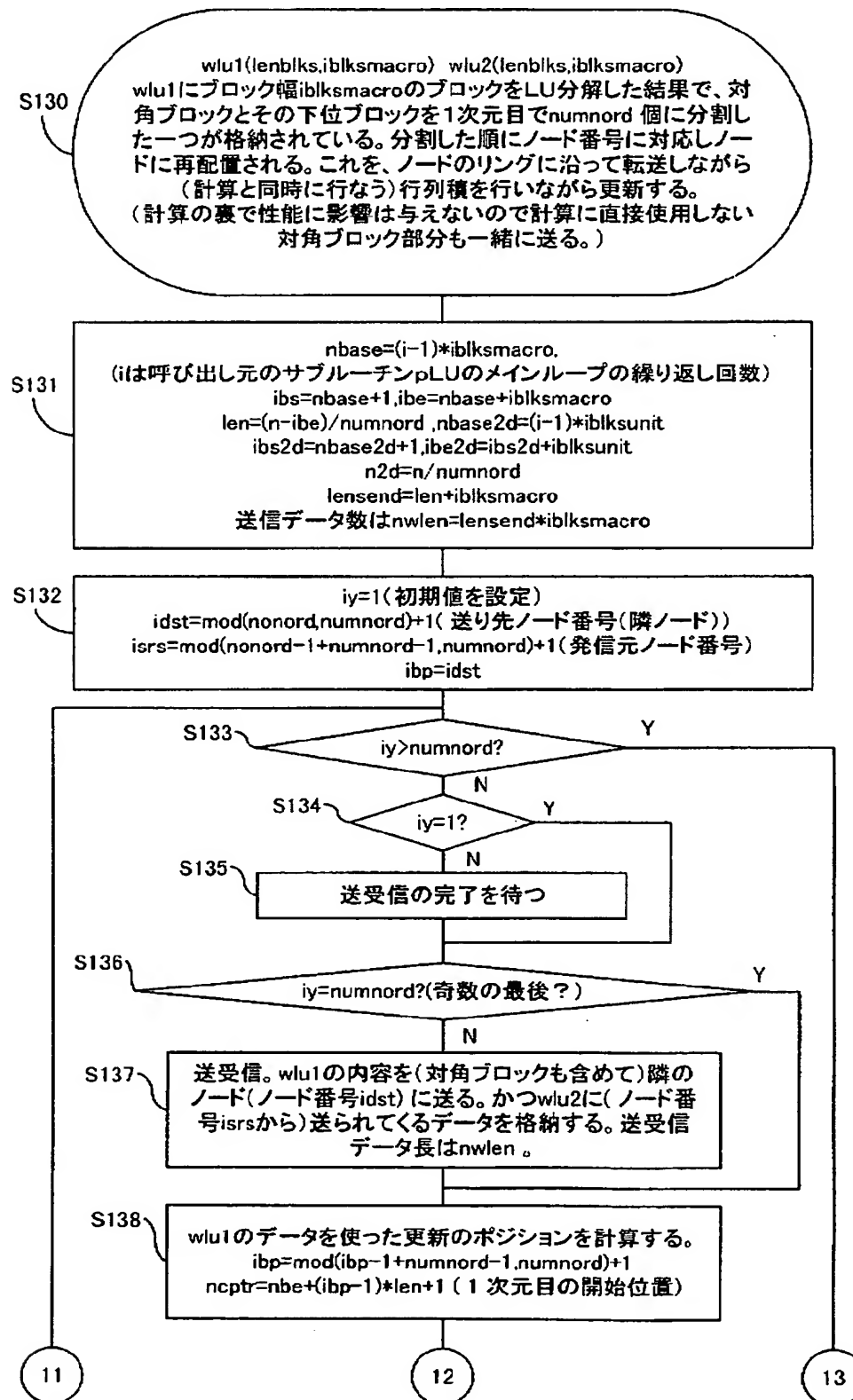
【図 21】

本発明の実施形態のフローチャート(その10)



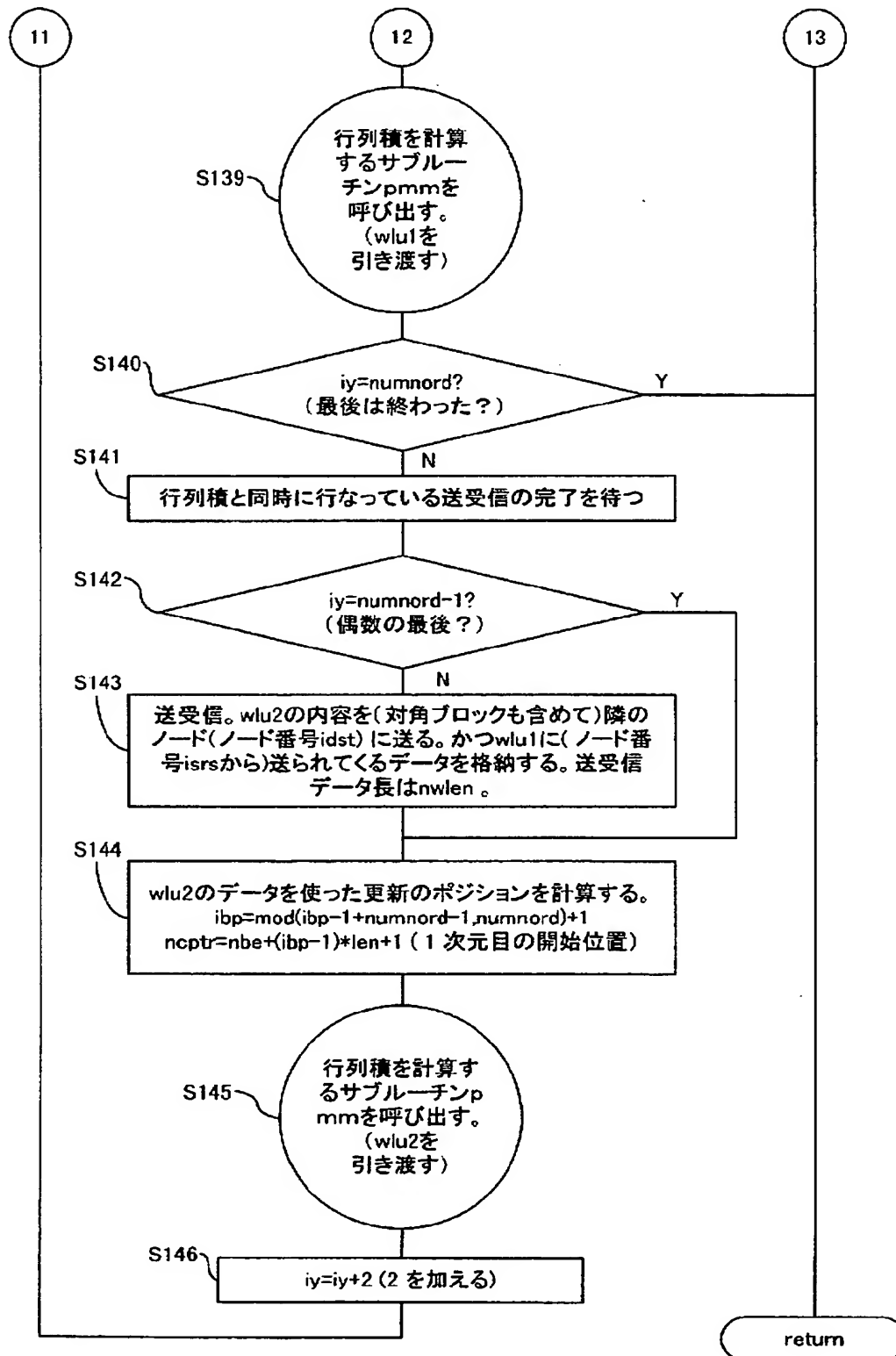
【図 22】

本発明の実施形態のフローチャート(その11)



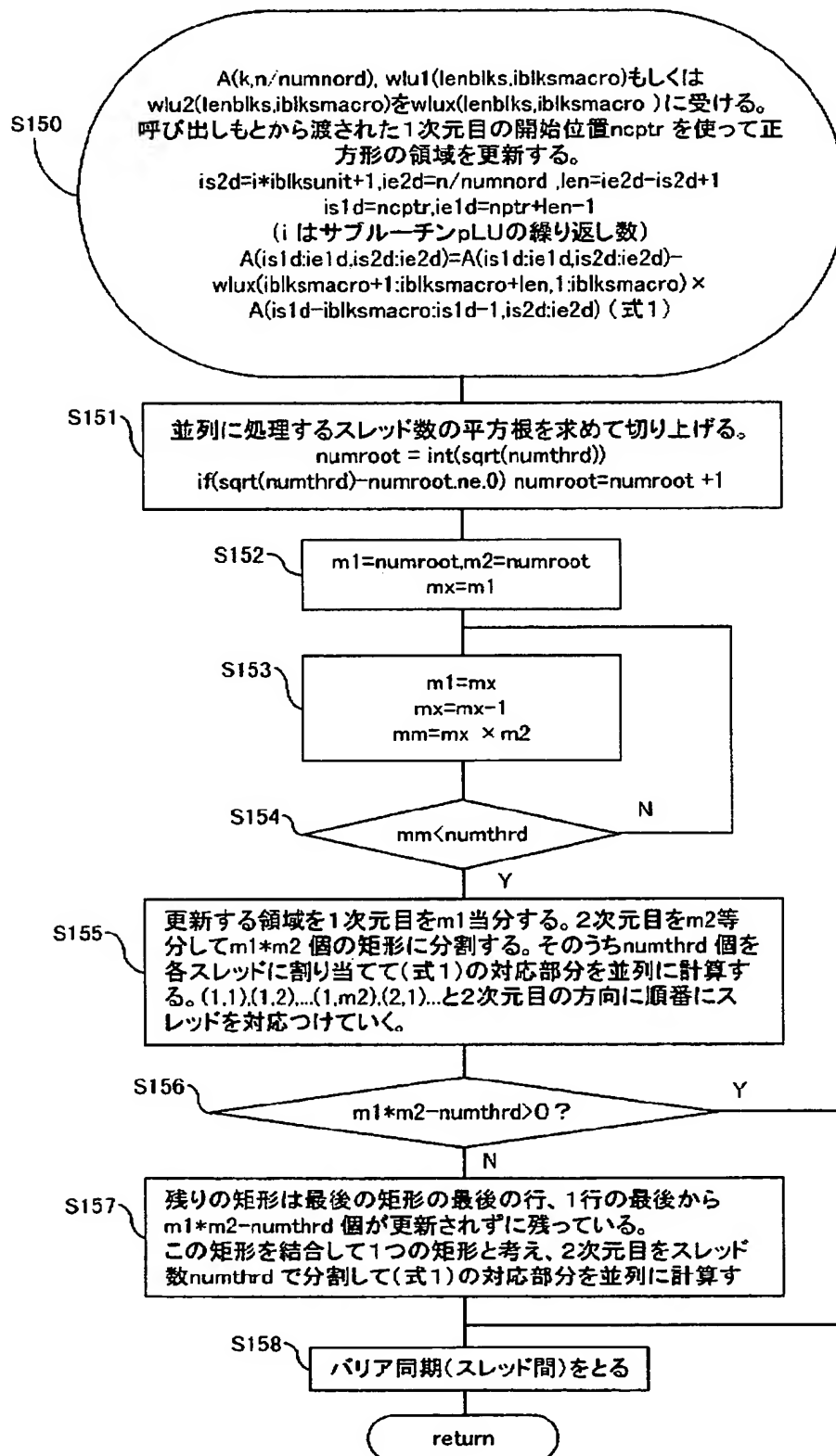
【図 23】

本発明の実施形態のフローチャート(その12)



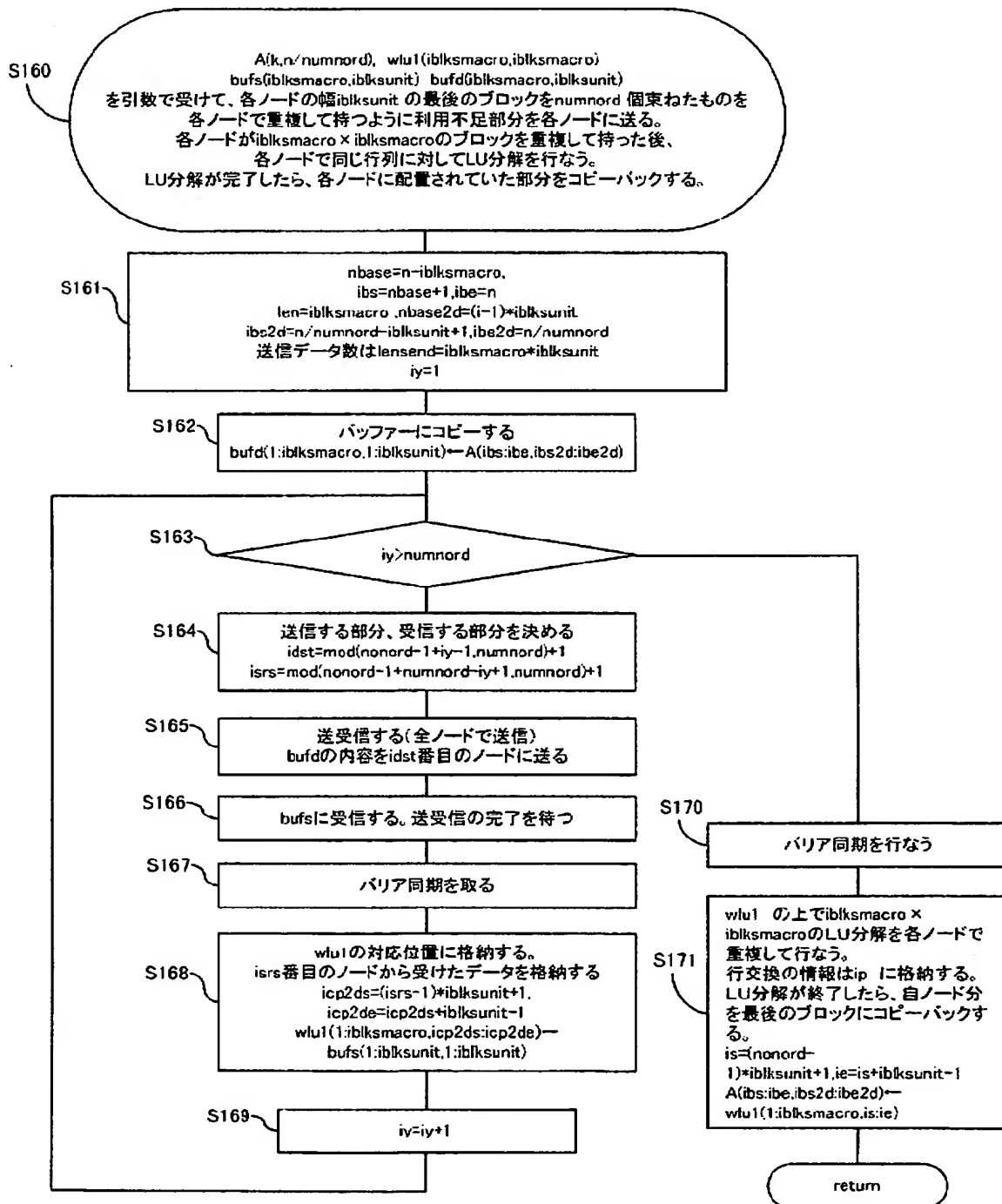
【図 24】

本発明の実施形態のフローチャート(その13)



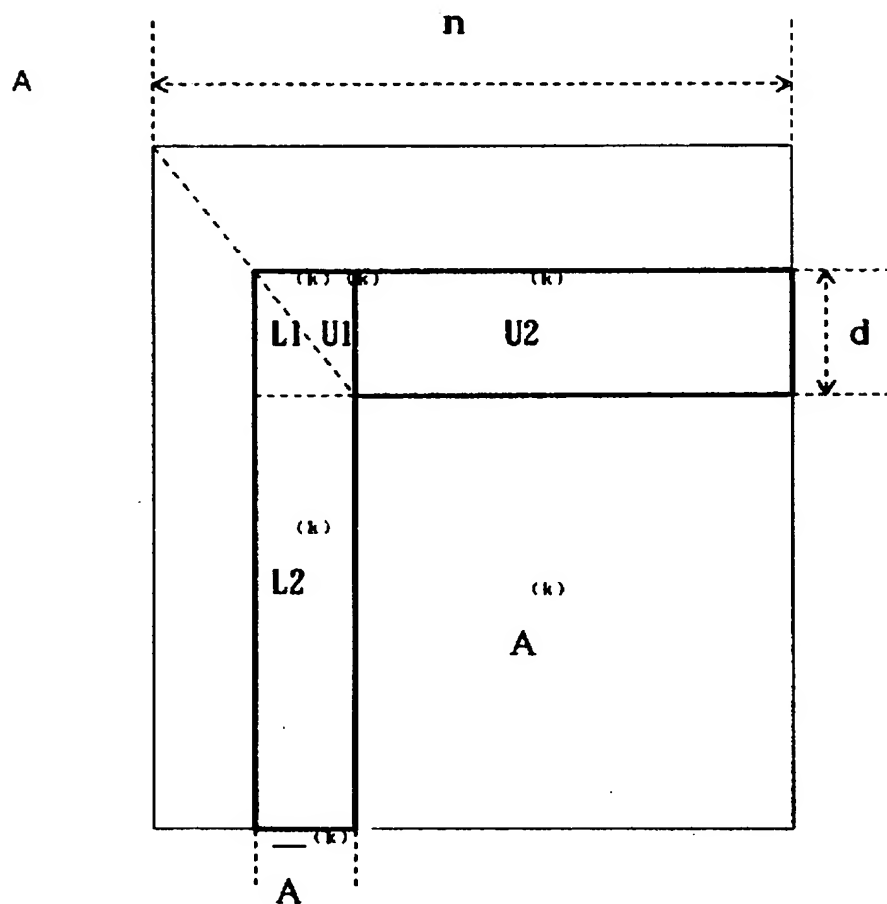
【図 25】

本発明の実施形態のフローチャート(その14)



【図 26】

スーパースカラ並列計算機用LU分解法の
アルゴリズムを概略説明する図



【書類名】 要約書

【要約】

【課題】 SMPノード分散メモリ型並列計算機で高速に行列を処理することの出来る装置あるいは方法を提供する。

【解決手段】 ブロック化された行列のLU分解法において、ネットワークで接続されたSMPノードのそれぞれに、行列の内、更新べきブロックを縦にノード数分に分割し、それぞれ分割された部分を各ノードに配置する。このような配置を更新すべきブロックについて、続けて行い、各ノードに、分割されたブロックの分割部分をサイクリックに割り当てるようにする。割り当てられた各ノードは、元のブロックの順番で分割されたブロックを更新する。元のブロックを順次更新することにより、各ノードの処理済みブロックの量が同じ分ずつ増えていき、負荷の分散をすることができる。

【選択図】 図4

特願 2003-095720

出願人履歴情報

識別番号

[000005223]

1. 変更年月日

1996年 3月26日

[変更理由]

住所変更

住 所

神奈川県川崎市中原区上小田中4丁目1番1号

氏 名

富士通株式会社